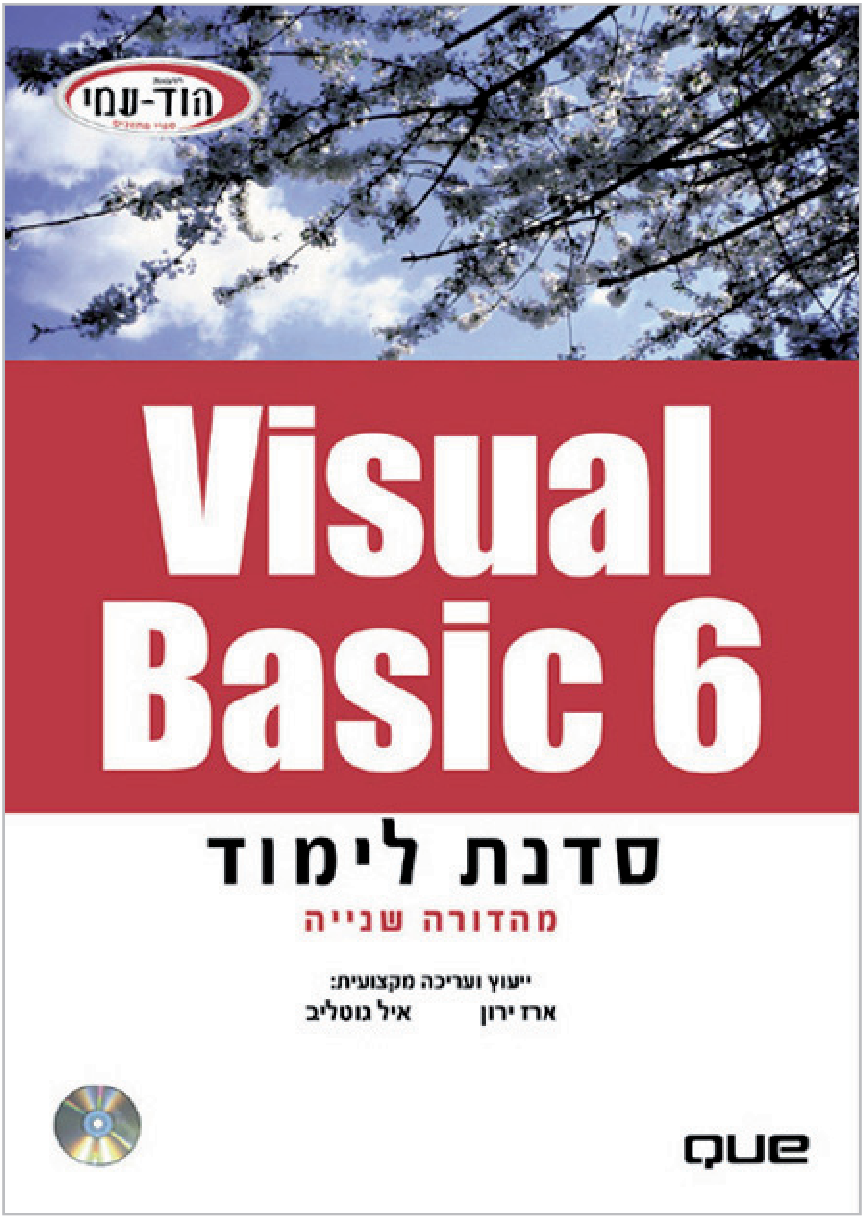


יש להתעלם מכל מה שנאמר לגבי תקליטור מצורף בספר זה

קבצי התוכניות (קוד מקור) נמצאים באתר הוד-עמי
www.hod-ami.co.il

בתיקיה "קבצי תרגול לספרים שלא באתר"

Visual Basic 6

סדנת לימוד

יש להתעלם מכל מה שנאמר לגבי תקליטור מצורף בספר זה

קבצי התוכניות (קוד מקור) נמצאים באתר הוד-עמי

www.hod-ami.co.il

בתיקיה "קבצי תרגול לספרים שלא באתר"

עורך ראשי: **זהר עמיהוד**

עריכה מקצועית, בדיקת תוכניות ותפיסת מסכים:

איל גוטליב - מנהל מרכז תמיכה מיקרוסופט

ארז ירון - מרכז תמיכה מיקרוסופט

Microsoft Certified Professional

עריכה מקצועית, לשונית ועיצוב: **שרה עמיהוד**

תרגום: **אס.אר.סי. בע"מ**

עיצוב עטיפה: **סטודיו מצגר**

תודה לכל הקוראים ששלחו את הערותיהם

לשם שטף הקריאה כתוב ספר זה בלשון זכר בלבד. ספר זה מיועד לגברים ונשים כאחד ואין בכוונתנו להפלות או לפגוע בציבור המשתמשים/ות.

- **טלפון: 09-9564716**
- **פקס: 09-9571582**
- **דואר אלקטרוני: Info@hod-ami.co.il**
- **אתר באינטרנט: http://www.hod-ami.co.il**

Visual Basic 6

סדנת לימוד

בריאן שילר, ג'ף ספוטס

que

הוצאת
הוד-עמי

Special Edition - Using Visual Basic 6

By **Brian Siler & Jeff Spotts**

Editor: **I. Amihud**

הוצאת הוד-עמי
לספרי מחשבים בע"מ
ת.ד. 6108 הרצליה 46160
טלפון: 09-9564716 פקס: 09-9571582
דואר אלקטרוני: info@hod-ami.co.il

הודפס בישראל 1999
מהדורה שנייה 1/2000

קדם דפוס: **טלאור טכנולוגיות**

דפוס: **דפוס איל**

ISBN 965-361-186-0 מסת"ב

תוכן עניינים מקוצר

חלק 5: נושאים מתקדמים בתכנות

חלק 6: Visual Basic ומסדי נתונים

חלק 7: Visual Basic והאינטרנט

חלק 8: נספחים

תוכן העניינים

פרק 3: אבני היסוד של Visual Basic..................71

פרק 4: שימוש בפקדי ברירת המחדל של Visual Basic 95

חלק 2: תכנות ב- Visual Basic

פרק 7: שימוש בתיבות דו-שיח לקבלת מידע 181

פרק 10: שליטה במהלך התוכנית 257

פרק 11: ניהול הפרויקט: תת-שגרות, פונקציות וריבוי טפסים 281

חלק 3: רכיבי תוכנית ב- Visual Basic

פרק 12: הפקדים המשותפים של Microsoft 305

פרק 13: עבודה עם מערכי פקדים........ 359

פרק 15: הרחבת פקדי ActiveX 417

פרק 16: מחלקות: שימוש חוזר ברכיבים 435

חלק 4: ממשקי Visual Basic

פרק 17: יישומים בעלי ממשק מרובה מסמכים.... 461

פרק 18: תכנון ממשק נכון 493

פרק 19: שימוש במרכיבי התכנון הוויזואלי......... 513

חלק 5: נושאים מתקדמים בתכנות

פרק 20: גישה ל-API של Windows................ 543

פרק 21: עבודה עם קבצים 567

פרק 22: שימוש ב-OLE לשליטה על יישומים אחרים 591

פרק 23: ארגז הכלים של המומחה 605

חלק 6: Visual Basic ומסדי נתונים

פרק 24: יסודות מסד הנתונים 635

פרק 25: פקד הנתונים ופקדי איגוד נתונים 657

פרק 26: שימוש באובייקטי גישה לנתונים (DAO) 681

פרק 27: גישה לאובייקטי נתונים מרוחקים (RDO) 717

פרק 28: גישה לאובייקטי נתונים באמצעות פקדי ActiveX (ADO) 735

חלק 7: Visual Basic והאינטרנט

פרק 30: שימוש ב-VBScript 797

פרק 31: מסמכי ActiveX 821

נספח 2: אריזת היישום 879

נספח 3: תקציר פקודות SQL 899

עבודות הייעוץ וההגהה הטכנית של ספר זה בוצעו על ידי שניים מהבולטים באנשי התמיכה בסביבת Microsoft בתקופה זו. השניים מבטאים את המגמות המתגבשות בתעשיית התוכנה המתחדשת ללא הרף.

איל גוטליב, הינו מנהל מרכז התמיכה הטלפוני של חברת Microsoft ישראל, והוא אחראי באופן יום-יומי על מספר רב של אנשי תמיכה מומחים במוצרי Microsoft. איל הינו בעל ניסיון רב בסביבת המשתמש והבנת צרכיו, בפיתוח בסביבת המשרד הממוחשב וכן בפתרון סוגיות הקשורות בהתאמת שגרות מערכת ההפעלה, עם דגש על הסביבה העברית.

איל שהגיע אל תחום המחשוב דרך שלל תפקידים שאינם קשורים כלל לענף, ממצה את היכולות הגלומות במוצרי הפיתוח המואץ (RAD) דוגמת Visual Basic, מוצרים המאפשרים לכל אדם, הפתוח לקליטת החשיבה התכנותית, לפתח יישומים מורכבים, תוך שימוש בטכנולוגיות שבקצה הקשת הטכנולוגית, הקפדה על יציבות, קלות פיתוח והתמקדות במרכיבים החשובים.

ארז ירון מייצג את המספר הגובר של מוחות צעירים ומבריקים בענף המחשבים. צעירים אלה, למרות גילם, מציגים יכולות מרשימות בתחום הטכנולוגיה המתקדמת.

ארז, שהינו מהראשונים שבמוסמכי Visual Basic 6 בישראל, מחזיק גם מספר תארי הסמכה נוספים של Microsoft במערכות הפעלה, ומשמש כאיש תמיכה בכיר במרכז התמיכה של Microsoft, וכמפתח פנים-ארגוני.

Rona Lustig
Premier Support for Developers
Microsoft Israel

פתח דבר

שפת הפיתוח Visual Basic, בשונה משפות פיתוח אחרות, מתמקדת במשתמש המפעיל את היישום, בעוד שנקודת המוצא של שפות פיתוח האחרות היא הלוגיקה של התהליך, או יעילות ניהול משאבי המחשב (למשל C).

המשתמש **חייב** להיות נקודת המוצא, אחרת הוא נשכח בתהליך. נקודת המוצא של Visual Basic היא: קודם כל נביט מנקודת ראותו, נבין מה הוא רוצה לראות וכיצד היה רוצה להפעיל את התוכנית, אותה אנו מפתחים, ורק אז נוסיף מאחורי הקלעים את הקוד המבצע זאת.

סביבת הפיתוח של Visual Basic באה לענות גם על צרכים נוספים של מפתחי תוכנה:

- פשטות כתיבת קוד - שפת התכנות מבוססת על Basic הישנה והפשוטה.
- קלות הפעלה - סרגלי כלים ותפריטים כמו בכל מוצרי Office (Word, Excel ו-PowerPoint).
- מערכי מידע העומדים לרשות ולעזרת המפתח - רק הקש F1 או לחץ לחיצה ימנית בעכבר.
- אשפים (Wizards), תבניות (Templates) ועזרים (AddIns) - קטעי תוכניות מוכנים מראש, לזירוז תהליכי פיתוח.
- שימוש ברכיבים (Components) מוכנים - במיטב הטכנולוגיה העכשווית: קח פקד, שים אותו בתוכנית שלך, וכאילו כתבת את כל קטעי הקוד שמאחוריו.
- יצירת רכיבים לשימוש מרובה - צור פקדים בעצמך, לשימוש תוכניותיך או כתוכנית קטנה לשימוש מפתחים אחרים.
- גישה קלה לבסיסי נתונים - בדיוק במידת המאמץ שברצונך להשקיע. תוכל לפתח תוכנית המציגה מידע השמור בבסיס נתונים (למשל Access), ומאפשרת לך להוסיף מידע, למחוק מידע ולשנות אותו, ללא כתיבת שורת קוד אחת!
- פיתוח מואץ ליצירת אב-טיפוסים - מינימום שורות קוד, מקסימום עזרים סביבתיים (אשפים, תבניות, השלמה אוטומטית), מידע מספריות רכיבים (Object Browser) ומאגרי עזרה גדולים, ברורים ועמוסים בדוגמאות קוד מוכנות...
- פתיחות מלאה למערכת ההפעלה - שימוש בשגרות מערכת ההפעלה (DLLs), לפעילויות מורכבות המצויות מחוץ לתחום הנתמך על ידי השפה הבסיסית.
- תמיכה מובנית בטכנולוגיות החדשות ביותר - COM, תוך שימוש ביכולות השלמה אוטומטית וה-Object Browser.

את יתרונות סביבת הפיתוח המתקדמת, מבית Microsoft, תוכל למצוא בספר **Visual Basic 6 - סדנת לימוד**. הספר כולל הנחיות ברורות, המלוות בתמונות מסכים והרבה קטעי תוכניות, אשר ילוו אותך החל מבניית יישום פשוט ועד לפיתוח יישומים מורכבים ומבוססי אינטרנט, תוך שימוש בטכנולוגיות המתקדמות ביותר. כל קורא והידע המוקדם שלו, והספר, בדיוק כמו Visual Basic, ייספק את ההדרכה הרצויה.

אך Visual Basic הינה יותר מסתם שפת תכנות וסביבת פיתוח. היא גם הבסיס ל-VBA (שפת הפיתוח המצויה מאחורי מוצרי Office ומוצרים רבים אחרים). VBA מאפשרת לך לכתוב קוד ב-Word, למשל, המקבל נתון ממשתמש, ועל פי נתון זה - מפעילה את Excel ליצירת גרף מנתונים השמורים ב-Access, וכל זאת בשורות קוד ספורות (לא יותר מ-10 פקודות), תוך ניצול היכולות הבנויות בתוך כל אחת ממערכות אלו, כל אחת בתחום מומחיותה.

ובל נשכח את האינטרנט. שם אנו פוגשים חבר נוסף במשפחת Visual Basic ושמו VBScript. ואם כבר היכרתם את VBScript - אתם יכולים לפתח גם טפסים ב-Outlook ואפילו תנועות מורכבות ב-SQL Server 7.0 !

מה עוד?

- פיתוח דפי DHTML - יש תבנית מוכנה.
- כתיבת IIS Applications - גם לכך יש תבנית.
- Reusable Code Components, OLE Servers, MTS Components.
- ל-Visual Basic פתרונות קלים, מהירים ואחידים לרוחב כל המערכת.

ביצועים ניתן לשפר על ידי שיפורים בחומרה (מעבד חזק יותר, הוספת זיכרון וכדומה). את נקודת ראותו של המשתמש ניתן לבטא בקלות, בבהירות ובמהירות בעזרת Visual Basic, והצעד הראשון מתחיל בעמוד הבא...

בהצלחה והנאה,

Rona Lustig

Premier Support for Developers

Microsoft Israel

מבוא

תכנות בסיסי ב-Visual Basic

ספר זה כולל אומנם חומר רב בנושאי תכנות מתקדמים ב- Visual Basic, אך בטרם תוכל להתמודד עם נושאים אלה עליך לרכוש תחילה יסודות תכנות מוצקים.

חלק I - היכרות ראשונית עם Visual Basic, יעניק לך היכרות ראשונית עם יכולות השפה ועם העבודה בסביבת הפיתוח.

- **פרק 1 - להתחיל ב- Visual Basic**, מסביר כיצד משמשת Visual Basic כלי ליצירת תוכניות מחשב מבוססות Windows, ומסייע בתכנון תוכניתך הראשונה.
- **פרק 2 - יצירת התוכנית הראשונה**, מלמד אותך, שלב אחר שלב, את מלאכת הבנייה של יישום המתפקד באופן מלא ב- Visual Basic.
- **פרק 3 - אבני היסוד של Visual Basic**, עורך עבורך סיור היכרות עם טפסים ופקדים, הרכיבים הבסיסיים בכל תוכנית אותה תיצור ב- Visual Basic.
- **פרק 4 - שימוש בפקדי ברירת המחדל של Visual Basic**, יילמד כיצד לעבוד עם קבוצת הפקדים השימושיים ביותר בעת עיצוב התוכנית ובנייתה.

חלק II - תכנות ב- Visual Basic, מכסה את כל היסודות להם אתה זקוק כדי שתוכל להתחיל בהרפתקת התכנות.

- **פרק 5 - תגובה באמצעות שגרות אירוע (Event Procedures)**, מתאר בפניך כיצד ניתן להקנות תכונות אינטרקטיביות לתוכנית Visual Basic. תלמד כיצד להקנות לתוכניתך יכולת תגובה לצעדים אפשריים שונים של משתמשיה.
- ב**פרק 6 - שליטה נוספת למשתמש: תפריטים וסרגלי כלים**, וב**פרק 7 - שימוש בתיבות דו-שיח לקבלת מידע** תלמד כיצד להרחיב את תוכניתך על ידי שימוש בתפריטים, סרגלי כלים ותיבות דו-שיח. באמצעות אלה ניתן ליצור ממשק המוכר למשתמש כתוצאה מעבודה קודמת עם אפליקציות מבוססות Windows. שימוש נכון בתפריטים, סרגלי כלים ותיבות דו-שיח מאפשר לך ליצור תוכניות מותאמות לאינטואיציה של המשתמש. במהלך פרקים אלה תבחן גם את **מודל האירועים** (Event Model) שהוא כאבן פינה בתכנות בסביבה מבוססת Windows.
- **פרק 8 - שימוש במשתנים לאחסון מידע**, ו**פרק 9 - יסודות התכנות של Visual Basic** מובילים אותך לתוך עולם פקודות, מבנים ומשתנים. Visual Basic מבוססת על יסודות מוצקים של שפת Basic. Visual Basic מועשרת במיגוון תכונות ופונקציות, ומאפשרת לכתוב תוכניות למשימות מכל סוג אפשרי. תלמד ליצור ולהשתמש במשתנים, ולבצע פעולות בשילוב עם מתמטיקה ומחרוזות.
- **פרק 10 - שליטה במהלך התוכנית**, מלמדך לשלוט בתוכניתך באמצעות שימוש בהחלטות (Decisions) ולולאות, ובאמצעות טיפול נכון בשגיאות.
- **פרק 11 - ניהול הפרויקט: תת-שגרות, פונקציות, וריבוי טפסים**, מעמיק את היכרותך עם יסודות התכנות, ומאפשר לך להרחיב את קוד המקור של תוכניתך.

עבודה עם רכיבי Visual Basic

חלק III - רכיבי תוכנית ב- Visual Basic, מרחיב ומעמיק את הידע הבסיסי שרכשת בפרקים קודמים ומציג בפניך רכיבים ממיגוון סוגים, אשר יאפשרו לך לשפר את איכות תוכניותיך. חשוב על רכיבים אלה כאבני בניין בעלות תכונות מתקדמות יותר מאלה הפשוטות, אשר היכרת בפרקים קודמים.

- **פרק 12 - הפקדים המשותפים של Microsoft**, מציג פקדים נוספים בהם תוכל להשתמש בעת בניית תוכניותיך ב- Visual Basic.
- **בפרק 13 - עבודה עם מערכי פקדים**, תלמד טכניקות מתקדמות יותר של שימוש בקבוצות משתנים, אשר יסייעו לך לייעל את תוכניותיך.
- **פרק 14 - יצירת פקדי ActiveX** ו**פרק 15 - הרחבת פקדי ActiveX** - יציגו בפניך את פקדי ActiveX בצורה מעמיקה. ניתן לעשות שימוש ברכיבים אלה במסגרת תוכניות Visual Basic, יישומי אינטרנט, או אפליקציות אחרות בעלות תמיכה ב-ActiveX. בפרקים אלה תלמד כיצד ליצור פקדי ActiveX וכיצד להרחיב את תכונותיהם לאחר יצירתם.
- **פרק 16 - מחלקות: שימוש חוזר ברכיבים**, מלמד אותך כיצד ליצור מחלקות ולנצל את יתרונות רכיבים רבי עוצמה אלה, בעיקר בעת עבודה עם מספר רב של תוכניות, המיועדות לביצוע משימות דומות.

יצירת ממשקי יישום

ב**חלק IV - ממשקי Visual Basic**, יוצגו בפניך מספר עקרונות אשר יסייעו לך בתכנון ממשקי היישום שלך. הממשקים הם "חלונות הראווה" של התוכנית, אשר דרכם מנוהלת האינטרקציה עם המשתמש.

- **פרק 17 - יישומים בעלי ממשק מרובה מסמכים** - ילמדך כיצד לבנות תוכנית העושה שימוש בממשק מרובה מסמכים (Multiple Document Interface-MDI). תוכנית MDI מאפשרת למשתמשיה לעבוד בו-זמנית עם מספר חלונות, המופיעים בשטח חלון ראשי אחד. תוכל להבחין בהבדלים שבין יישומי MDI ליישומים בעלי ממשקים המבוססים על מסמכים בודדים (Single Document Interface - SDI).
- ב**פרק 18 - תכנון ממשק נכון** וב**פרק 19 - שימוש במרכיבי התכנון הוויזואלי**, תלמד מדוע חשוב לתכנן את תוכניותיך בטרם תיגש לכתיבת הקוד. במסגרת פרקים אלה יוצגו בפניך צעדים שגויים העלולים לפגוע באיכות הממשק, ותלמד כיצד להימנע מטעויות פיתוח נפוצות.

תכנות מתקדם ב- Visual Basic

חלק V - נושאים מתקדמים בתכנות, כולל כמה וכמה נושאים החורגים מן החומר שלמדת עד כה, ומתמקד בטכניקות פיתוח ייחודיות.

- **פרק 20 - גישה ל- API של Windows**, מראה לך כיצד ניתן להשתמש בשגרות התכנות המוגדרות מראש, אשר משולבות, בהמוניהן, בממשקי היישומים של Windows. תלמד כיצד לנצל לצרכיך את ספריות הקוד העצומות, המשולבות בכל יישום מבוסס Windows.
- **פרק 21 - עבודה עם קבצים**, ילמדך כיצד להשתמש בקבצים מסוגים שונים למטרות אחסון ואחזור מידע.
- **פרק 22 - שימוש ב-OLE לשליטה על יישומים אחרים**, מתאר כיצד ניתן להשתמש ב-OLE על מנת ליצור אינטראקציה בין היישום שלך ליישומים אחרים, תוך ניצול מקסימאלי של יתרונות OLE.
- **פרק 23 - ארגז הכלים של המומחה**, מציג בפניך מספר אפליקציות עצמאיות של Visual Basic, המיועדות למיגוון משימות מיוחדות ומעניינות.

טכניקות לתכנות בסיסי נתונים

חלק נרחב מן התוכניות הקיימות בעולם העסקי בו אנו חיים, הן תוכניות המשלבות עבודה מול בסיסי נתונים. מידת מורכבותן של תוכניות אלו משתנה: יש תוכניות פשוטות, המיועדות לטיפול ברשימות דואר, ולעומתן תוכניות מורכבות המיועדות לטיפול במערכת הזמנות וחיובים בארגונים גדולים.

חלק VI - Visual Basic ומסדי נתונים, מציג בפניך את תהליך בנייתן של תוכניות בסיסי נתונים הנותנות מענה למיגוון צרכים.

- **פרק 24 - יסודות מסד הנתונים**, מתאר כיצד ליצור, לעדכן ולשנות בסיסי נתונים. **פרק 25 - פקד הנתונים ופקדי איגוד נתונים** ממשיך איתך צעד נוסף קדימה ומלמדך כיצד ניתן ליצור יישומים במהירות, על ידי שימוש משולב בפקד הנתונים והפקדים אשר ניתן לקשר אליו.
- **פרק 26 - שימוש באובייקטי גישה לנתונים (DAO)**, ו**פרק 27 - גישה לאובייקטי נתונים מרוחקים (RDO)**, ילמדו אותך כיצד לרתום את משאבי הפיתוח רבי העוצמה של Visual Basic, ליצירת אפליקציות בבסיסי נתונים.
- **פרק 28 - גישה לאובייקטי נתונים באמצעות פקדי ActiveX (ADO)** מתאר עבורך את המודל העדכני של Microsoft לעבודה מול בסיסי נתונים.
- ובסוף, **פרק 29 - יצירת דוחות** מציג בפניך זוג כלי דיווח צד שלישי: Data Report Designer ו- Crystal Reports, המשמשים אותך להצגת מידע שמקורו בבסיסי הנתונים המנוהלים על ידי יישומיך.

תכנות מבוסס אינטרנט

בחלק VII - Visual Basic והאינטרנט, הנך נחשף לעולם מתפתח של שימוש בווריאציות Visual Basic לשם בניית אפליקציות מבוססות אינטרנט.

- **בפרק 30 - שימוש ב-VBScript** תלמד להכיר את קרוב המשפחה של Visual Basic.
- **פרק 31 - מסמכי ActiveX** מרחיב ומגבש את הידע שצברת בפרקים קודמים לכדי יכולת פיתוח של רכיבי ActiveX, המאפשרים שיתוף בשיטה מתקדמת.
- **פרק 32 - Visual Basic ושימושים נוספים באינטרנט**, מלמד כיצד לשלב תכונות אינטרנט ביישום, כדוגמת יכולות דפדוף בדפי הרשת וניהול דואר אלקטרוני.

התייחסויות נוספות

בחלק VIII - נספחים תוכל להסתייע כמדריך מקצועי בעת תכנון ופיתוח יישומים ב- Visual Basic, כמו כן תמצא שם הוראות על אופן השימוש בתקליטור.

- **נספח 1 - הכרת סביבת הפיתוח**, לוקח אותך לסיור בממשק המשתמש של Visual Basic. תוכל להיעזר בו כמדריך במהלך היכרותך הראשונית לסביבת הפיתוח.
- **נספח 2 - אריזת היישום**, מתאר בפניך כיצד להכין את תוכניות Visual Basic להפצה.
- **נספח 3 - תקציר פקודות SQL**, מניח בפניך את יסודות שפת SQL (Structured Query Language), המשמשת לביצוע משימות כגון אחזור מידע בבסיסי נתונים. תוכל להפיק תועלת מן המידע בנספח במהלך קריאת הפרקים העוסקים בגישה לבסיסי נתונים באמצעות תוכניות Visual Basic .
- **נספח 4 - דפי שרת פעילים**, מלמד כיצד VBScript מעובד על שרתי אינטרנט, כדי ליצור גמישות במידע המוצג ברשת.
- **נספח 5 - התקליטור המצורף**, פרטים בהמשך.

תוכניות וקוד מקור בהם נעשה שימוש בספר

כל קטעי הקוד המופיעים בספר זה, ניתנים בתקליטור המצורף. תוכל להשתמש בקטעים אלה בשלמותם כדי לחסוך זמן, ולמנוע טעויות הקלדה במהלך העבודה עם הדוגמאות שבספר. כמו כן, תוכל למצוא את כל התוכניות המופיעות בספר.

התקליטור המצורף -

בתקליטור המצורף תמצא את תיקיית קבצי הקוד הרלוונטיים לספר זה - תיקיה בשם **59221** הנמצאת תחת התיקיה **Books**. בנוסף, תמצא בו קטלוג HTML של **הוד-עמי** ותוכנות חופשיות. להרחבה על תכולת התקליטור ואופן השימוש בו, פנה לנספח 5.

מוסכמות ואלמנטים מיוחדים בהם נעשה שימוש בספר זה

ספר זה עושה שימוש במספר מוסכמות ואלמנטים מיוחדים, במטרה להדגיש נושאים מסוימים, ולהקל על הקריאה והשימוש בספר. בטרם תתחיל בקריאה, מומלץ כי תתבונן במוסכמות ואלמנטים אלה ותאפשר להם להרחיב את ניסיונך כקורא:

מוסכמות

הרשימה שלהלן מפרטת את המוסכמות בהן עושה הספר שימוש:

- מילים ומשפטים המדגישים את כוונת המחבר, כמו גם מונחים חדשים, מופיעים **באותיות מודגשות**.
- קטעי הקוד מופיעים ברקע אפור.
- מקשי הפעלה מסומנים על ידי קו מדגיש, למשל, הביטוי הבא: **Tools, Options** מלמד כי ניתן להיכנס לתפריט Tools על ידי צירוף המקשים Alt+T ואחר להפעיל את הפקודה Options על ידי הצירוף Alt+O.
- חלקים שונים בקטעי הקוד המובאים בספר, מופיעים מדי פעם *בגופן נטוי*. אלה הם חלקים אשר עליך להוסיף לתוכניות קיימות. שיטה זו מאפשרת לך לראות מה עליך להוסיף או לשנות על פי ההקשר.
- בשל מגבלות הרוחב של עמודי הספר, נאלצנו במקרים מסוימים, לשבור שורות קוד ארוכות במיוחד למספר חלקים. במקרים אלה, מופיע, בסוף השורה השבורה, סימן ההמשך (_) של Visual Basic. בעת העתקת הקוד למחשב, תוכל להשמיט את סימני ההמשך ולכתוב את הקוד במלואו בשורה בודדת או לחילופין - להעתיקו כפי שהוא מופיע בספר. תוכנת Visual Basic תדע לפרש את הקוד בכל אחת מן הצורות שתבחר.

אלמנטים מיוחדים

סוגים שונים של מידע מוצגים בספר באמצעות אלמנטים מיוחדים. להלן הסברים ודוגמאות אשר יעזרו לך להכיר את האלמנטים השונים ומשמעותם.

כל פרק מתחיל עם "מפת דרכים" או רשימת נושאי משנה. כמו כן, בסוף כל פרק, תמצא סעיף "מכאן והלאה..." בו תמצא סיכום קצר של הנושאים שנדונו במהלך הפרק וכן הפניות לפרקים אחרים, אשר עוסקים בנושאים קשורים, או מרחיבים את הדיון בנושאים שכבר למדת.

ראה הפניות להערות הסימוכין המשולבות בגוף הפרק, והפניות למקומות נוספים המכילים מידע לגבי הנושאים הנלמדים.



טיפ:

המלצה קצרה ושימושית המתייחסת לפעולות ותהליכים מהירים הנוטים לחמוק מן העין במהלך התכנות.



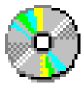
הערה:

מידע נוסף אשר עשוי לעזור לך להימנע מבעיות מיותרות, או עצה טובה הקשורה לנושא הנלמד.



אזהרה:

אזהרה בפני בעיות העלולות להיגרם על ידי תהליך מסוים, תוצאות בלתי צפויות וטעויות נפוצות שיש להימנע מהן.



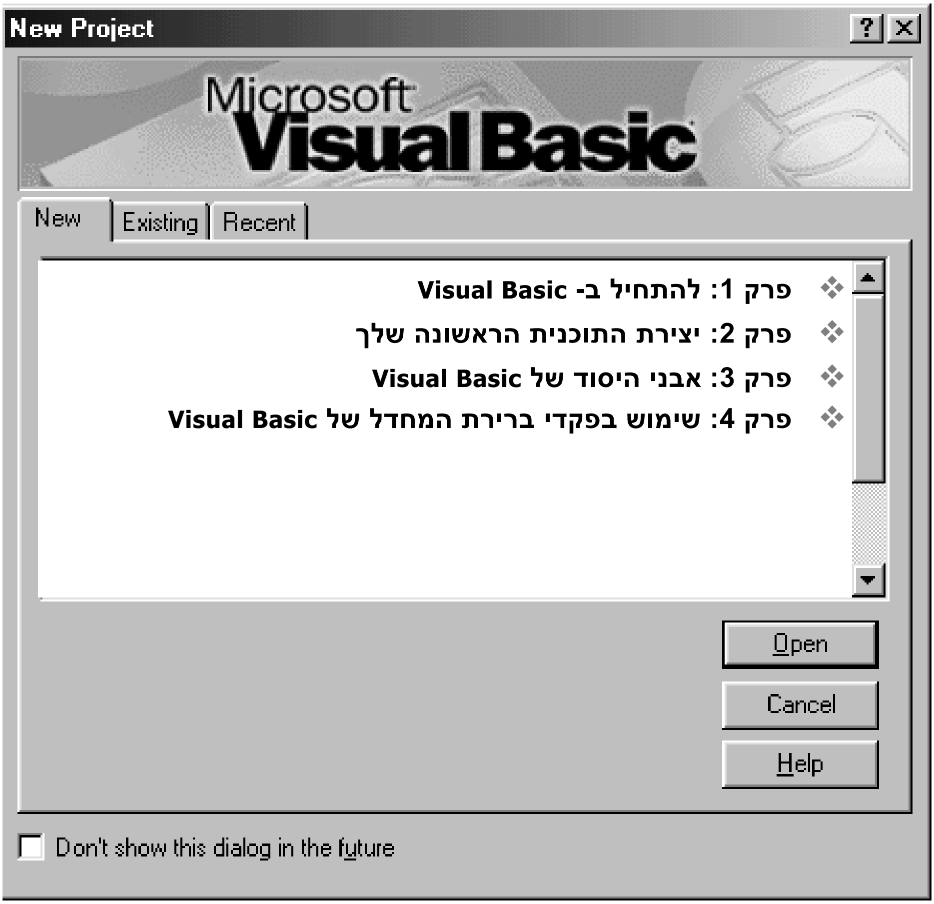
בתקליטור:

הפניות לקבצי הקוד שבתקליטור המצורף.

הערות שוליים: הסברים ארוכים יותר אשר אינם חלק מן הזרימה השוטפת של הפרק. קרא את הערות השוליים לצורך העמקת היכרותך עם הנושאים הנלמדים.

חלק 1

היכרות ראשונית עם Visual Basic



1

להתחיל ב- Visual Basic

בפרק זה:

- מהי תוכנית מחשב
- תוכניות מחשב ושפות תכנות
- Visual Basic כשפה חכמה
- החשיבות הטמונה בתכנון מוקדם של תוכניתך

כאשר אנשים פוגשים אדם המומחה ב- Visual Basic הם בדרך כלל מעלים בפניו את השאלות הבאות:

- מהי Visual Basic?
- האם Visual Basic היא תוכנית?
- מה ניתן לעשות עם Visual Basic?
- מדוע כדאי לי ללמוד Visual Basic?

חבילת Visual Basic של חברת Microsoft מוגדרת כ**מערכת פיתוח**. באופן פשטני, מערכת פיתוח זו מיועדת לכתיבת יישומי מחשב מבוססי Windows. בחבילה נכללים Visual Basic ומספר כלים נוספים המיועדים לסייע בכתיבת תוכניות. Visual Basic, כשלעצמה, אינה אפליקציה, אלא כלי שבאמצעותו ניתן לפתח אפליקציות.

חבילת Visual Basic מאפשרת לפתח יישומים המותאמים באופן אישי לדרישות המשתמש ופותרת אותך (או את החברה שלך) מהתלות במוצרי מדף סטנדרטיים. תוכנית מחשב טובה היא זו המגלה גמישות ומתאימה את עצמה לדרישות המשתמש, זאת בניגוד לתוכנית המאלצת את המשתמש להתאים את עצמו למגבלותיה.

פרק זה מציג בפניך את משמעות המושג **תוכנית מחשב** ודן בשפות תכנות ובתרומתן ליצירת תוכניות המותאמות לצרכיך. הפרק מסיים בציון מספר נקודות חשובות אשר עליך להביא בחשבון בעת תכנון וגיבוש תוכנית מחשב. חלק זה כולל גם קישור קצר לפרק 2, ***יצירת התוכנית הראשונה***.

מהי תוכנית מחשב?

תוכניות המחשב אשר אתה מפתח ב- Visual Basic מכונות גם **יישומים**, **תוכנות** או **אפליקציות**. תוכניות אלו יכולות לבצע כמעט כל משימה אשר אתה מעלה בדמיונך. יישומי מחשב מתחלקים בדרך כלל, לשתי קטגוריות עיקריות: **מוצרי מדף** (Packaged), ו**יישומים מותאמים אישית** (Custom).

המונח **מוצרי מדף** מתייחס לכל אותם יישומים אשר ניתן לרכוש בחנויות התוכנה, באמצעות משלוח בדואר, ישירות מן היצרן וכיוצא בזה. מוצרי מדף מתוכננים מראש לטיפול במשימות מסוימות. לדוגמה, ניתן לרכוש את התוכנה Microsoft Word לעריכה ועיבוד תמלילים, את התוכנה Win Fax של Symantec לשליחה וקבלת פקסים באמצעות המחשב, או את תוכנת VirusSacn של MCAfee להגנה על המחשב מפני וירוסים. לקטגוריה זו משתייכים גם משחקי המחשב, כדוגמת Myst של Broderbund או Flight Simulator של Microsoft. היישומים שהוזכרו לעיל הינם חלק מזערי מעשרות אלפי מוצרי המדף המיועדים למחשבי PC מבוססי Windows.

יישומים מותאמים אישית מיועדים בדרך כלל לביצוע משימה מיוחדת כלשהי במסגרת ארגון מסוים. לדוגמה, חברה מסוימת הזקוקה לתוכנית מחשב אשר תנהל מערכת הזמנות ותנהל מעקב אחר מוצרים, מרגע כניסתם למלאי ועד שיווקם ללקוח. אם לא קיים בשוק מוצר מדף העונה לצרכי החברה, ניתן לפתח יישום מותאם אישית,

לביצוע המשימה הדרושה. אחד היתרונות ביישום מותאם אישית במקרה זה, היא שניתן להתאימו באופן שוטף לצרכי החברה המשתנים. זאת להבדיל ממוצרי מדף, אשר בדרך כלל לא ניתן להתאימם או לשנותם על פי צרכיו האישיים של המשתמש.

הערה:

יש תוכניות מחשב, המהוות שילוב של מוצר מדף ויישום מותאם אישית. יצרני תוכנה שונים פיתחו מוצרי מדף המכילים תכונות אשר המשתמש יכול להתאים לצרכיו. תופעה זו נפוצה בעיקר אצל יצרני תוכנה של שוק אנכי, אשר מוצריהם מיועדים לתחום תעשייתי מסוים. לדוגמה, כשלעצמי, השקעתי מספר שנים בפיתוח תוכנה המיועדת למנהלי תערוכות ומסייעת בארגון שווקי חוצות, ירידי עתיקות, אירועים מסחריים וכיוצא בזה. כל לקוחותיי משתמשים במערכות זהות ביסודן, אשר עברו התאמות ושינויים כדי לענות לצרכיהם האישיים, בהתאם לסיטואציות הארגוניות השונות.

סוג נוסף של תוכנות "בנות תערובת" הינם מוצרי מדף בעלי ממשק משתמש גמיש. ממשקים אלה, הניתנים להתאמה אישית על ידי המשתמש, הופכים למעשה, כל מוצר מדף ליישום מותאם אישית!

לכל תוכניות המחשב, באשר הן, קיים מכנה משותף בסיסי והוא עובדת היותן כתובות בשפת תכנות אחת או יותר, כדוגמת Visual Basic.

תוכניות מחשב ושפות תכנות

תוכנית מחשב היא בסופו של דבר סדרת הוראות הגורמות למחשב לבצע משימה מסוימת. שפת מחשב, כדוגמת Visual Basic, משמשת לתרגום הוראות משפת בני האדם לפקודות אותן יכול המחשב להבין ולבצע.

אם נביט לרגע ברמה הבסיסית ביותר של המחשב, נמצא כי המעבד, האחראי יותר מכל על ביצועי המחשב, מבין אך ורק הוראות ספרתיות, או ליתר דיוק - פקודות פשוטות ביותר העוסקות בעיקר בהעברות מספרים בין כתובות זיכרון. פקודות פשוטות אלה, אותן מבין המעבד, נקראות **שפת מכונה** (Machine Language), כלומר הפקודות הבסיסיות ביותר באמצעותן ניתן לתפעל את מחשב PC.

מקובל להתייחס לשפת מכונה כשפת **רמה נמוכה** (Low Level Language), זאת משום שהיא ממוקמת עמוק למטה, ברמת ההבנה של המעבד. כפי שאתה בודאי מנחש, כתיבת תוכניות בשפת מכונה היא משימה מרתיעה למדי. למזלנו, לא נדרשת בקיאות בשפת המעבד לצורך בניית תוכנית מחשב. כיום, עומדות לרשותנו מיגוון שפות תכנות, המאפשרות לנו לעבוד ברמות גבוהות יותר ולכתוב תוכניות המורכבות מהוראות מחשב בשפה הדומה מעט לאנגלית. בעת הפעלת התוכנית, ההוראות השונות מתורגמות לסדרת פקודות בשפת מכונה, אותן יכול המעבד להבין ולבצע.

בין שפות התכנות שהתפתחו במרוצת השנים, ניתן לציין את Fortran ו-Cobol, המשמשות בעיקר מפתחים העובדים על מערכות מחשב מרכזיות ולעומתן - Basic, Pascal, C ו- ++C המשמשות בדרך כלל, לכתיבת תוכניות ברמת המחשב האישי.

Visual Basic קיימת בשוק מספר שנים ואפשר לראותה כתולדה של Beginner's) BASIC All-Purpose Symbolic Instruction Code) אשר התפתחה, כפי שמעיד שמה, כשפה המיועדת לתוכניתנים מתחילים. מפתחים רבים רכשו את ניסיונם המקצועי הראשון בשפת BASIC, בטרם המשיכו לשפות בעלות עצמה רבה יותר.

Visual Basic, אשר פותחה על ידי Microsoft לאחר פיתוח Windows, היא למעשה גירסה ויזואלית של BASIC. מאז שהוצאה לשוק, התפתחה Visual Basic במהירות והפכה לכלי פיתוח רב עצמה, והשאירה הרחק מאחור את המוניטין של קודמתה, כשפה המיועדת למתחילים בלבד.

אחת התכונות הנחמדות של Visual Basic היא העובדה שניתן לפתח באמצעותה יישומים יציבים במהירות רבה. כפי שתיווכח במהלך לימודיך, Visual Basic חזקה מאוד בקיצור תהליכי פיתוח, אשר בדרך כלל גוזלים מן המפתח זמן ואנרגיה רבים. תכונה חשובה זו חוסכת ממנו משימות פיתוח אפרוריות ושגרתיות ומפנה את זמנו לשיפור איכותה ורמת ביצועיה של התוכנית. Visual Basic נחשבת לכלי מהיר לפיתוח יישומים, או בשפה מקצועית - **RAD** (Rapid Application Development Tool).

Visual Basic היא שפה חכמה

הסיבה העיקרית לפופולריות ולעוצמה של Visual Basic, זהה לזו העומדת ביסוד הצלחתה של Windows. חברת Microsoft לקחה טכנולוגיה מורכבת (פיתוח יישומי מחשב) ופישטה אותה באופן משמעותי באמצעות ממשק עבודה גרפי. נניח כי עליך לכתוב תוכנית עבור החברה בה אתה עובד. סביבת פיתוח ויזואלית מאפשרת לך לעצב את ממשק המשתמש במהירות רבה תוך ארגון החלונות והאלמנטים השונים על גבי המסך, באופן הדומה לסידור גיליון עיתון. בכלי פיתוח מבוססי טקסט, לעומת זאת, אתה שולט בממשק המשתמש באמצעות תוכנית של פקודות כתובות. ההגיון מלמד כי קל יותר לרכוש יכולת תכנות בשפות הוויזואליות, וכי העבודה בסביבה גרפית חוסכת זמן רב. במקרה זה, נכונה הסיסמה הישנה "תמונה אחת שווה אלף מילים".

עם זאת, אל תטעה לחשוב ש- Visual Basic היא רק עוד סוג של ממשק יפה. עקרון יסוד נוסף של Visual Basic היא היכולת להשתמש ברכיבים, או **עצמים** (Objects) המובנים בה מבפנים. סוג של עצמים עליהם תלמד בקרוב מאוד, הם ה**פקדים** (Controls) של Visual Basic.

הפקדים המזכירים מאוד רכיבים בלוח המחוונים ברכב, הם אלמנטים המשמשים בעת יצירת ממשק המשתמש. ניתן להשתמש בהם להצגת מידע (כפי שעושה מד המהירות), או לביצוע פעולה מסוימת (כפי שעושה המתנע). רבים מהתהליכים המתרחשים ברכב, כגון שילוב בין הצתה-מתנע ומנוע, נסתרים מהעין. הנהג מבחינתו שולט בפעולות הרכב באמצעות לוח המחוונים. באופן דומה, פקדי Visual Basic מאפשרים להוסיף תכונות שונות, מבלי שתצטרך להיות מעורב באופן פעולתן "מאחורי הקלעים". לדוגמה, אם ברצונך שתוכניתך תדע לקלוט מידע, כל שעליך לעשות הוא לשבץ פקד מתאים. זה למעשה היתרון הגדול של שפות תכנות ויזואליות, תוכל להתרכז במשימות התכנות ולא כיצד לגרום לשפת התכנות לבצע אותן.

החשיבות הטמונה בעיצוב מוקדם של תוכניתך

פרופסור שלימד אותי אנגלית בקולג' תיאר בפנינו פעם את סוגי הסופרים השונים: "סופר מסורתי בדרך כלל כותב את הסיפור שלו במבנה של התחלה - אמצע - סוף. מודרניסט עשוי לשמר את אותו מבנה ופוסט מודרניסט יכלול רק שניים מבין השלושה". למרבה הצער, מחברי תוכניות מחשב אינם נהנים מאותה פריבילגיה. הואיל ועבודת התכנות היא מובנית מטבעה, קיימת חשיבות רבה לתכנון מוקדם של התוכנית **בטרם** ניגשים לכתיבתה בפועל. אמנם, מתכנתים רבים נוטים להתחיל מייד בכתיבת הקוד אך מומלץ בחום כי תסגל לעצמך הרגל של תכנון מוקדם, אפילו בתוכניות קטנות ופשוטות. לכל הפחות, שב מול דף נייר ריק, רשום לעצמך הערות לגבי המשימות שאתה מייעד לתוכנית וצייר סקיצה של ממשק המשתמש.

הערה:

יש לציין כי האסטרטגיה אשר אנו מציגים בפרק זה, מהווה מערכת כללית של הנחיות ואינה מתיימרת, בשום אופן, להיות הגישה הנכונה היחידה לתכנות. הטכניקה בה תבחר להשתמש בסופו של דבר, עשויה להיות שונה לגמרי.

שילוב העיצוב המוקדם בתהליך הפיתוח

בעת בניית תוכנית מחשב, חשוב לנקוט בגישה מובנית. צעדים מסוימים צריכים להתבצע בסדר קבוע ומוטב שתתרגל, כבר מן ההתחלה, לעבוד בסדר הנכון. אמנם, מפתה מאוד להריץ Visual Basic ולגשת מייד למלאכה, אך האמן לנו - השקעת מאמץ קטן בעיצוב מוקדם, תחסוך לך זמן וכאבי ראש רבים בהמשך עבודתך.

להלן הצעדים שעליך לבצע במהלך בניית תוכנית מחשב:

1. תכנון משימות התוכנית (כיצד עליה לעבוד).
2. עיצוב ממשק המשתמש (כיצד עליה להיראות).
3. כתיבת הקוד של התוכנית (הוצאה לפועל של צעדים 1 ו-2).
4. בחינת ביצועי התוכנית וניפוי שגיאות (כולל בדיקת ביתא המבוצעת, במידת האפשר, על ידי משתמשים מחוץ לצוות הפיתוח).
5. תיעוד והפצה של התוכנית (שימוש).

צעדים אלה כוללניים ביותר ואינם משקפים את התהליך בשלמותו. בעת הצגת תוכנית הדוגמה בפרק הבא, נתאר מספר צעדים נוספים, המיוחדים לתכנות ב- Visual Basic.

שני הצעדים הראשונים המוצגים לעיל, משקפים את תהליך הבנייה בפועל של התוכנית. תוכנית הדוגמה המוצגת בפרק 2 מהווה בסיס טוב להדגמת הצעדים השונים הכרוכים בתכנון, בנייה והפעלה של תוכנית Visual Basic. שני הצעדים האחרונים

(בחינת ביצועים וניפוי שגיאות, הפצה) מסוקרים רק בשלבים מאוחרים יותר בספר, זאת למרות שאתה לומד כיצד להפעיל את התוכנית כבר במסגרת פרק 2.

הערה:

בעת פיתוח פרויקט רחב היקף, מומלץ, משיקולי נוחיות, לפרקו למספר חלקים ולטפל בכל חלק בנפרד. ניתן להסתייע בתכונות רבות של Visual Basic לצורך חלוקת העבודה על פרויקט גדול בין חברי צוות הפיתוח.

מעט על עיצוב התוכנית

בעת חניכת פרויקט חדש, מפתה מאוד להתחיל מייד בשרבוט הקוד. אחרי הכל, החלקים היצירתיים ביותר בתהליך הפיתוח הם שרטוט הממשק וכתיבת הקוד עבור התוכנית. עם זאת, תוכנית טובה מתחילה בעיצוב מוקדם. כשמדובר בפרויקטים קטנים, אין צורך בתרשים זרימה מפורט, אם כי מצד שני, גם בפרויקטים מסוג זה כדאי שתהיה תוכנית מוקדמת.

בתום תהליך התכנון מומלץ כי יימצאו בידך הפריטים הבאים:

- רשימה תמציתית של משימות התוכנית.
- לוח זמנים להשלמת כל משימה ומשימה.
- הבהרה של מידת התלות ההדדית בין חלקי התוכנית השונים.
- רשימת הקריטריונים על פיהם תיבדק התוכנית.

לתוכנית בעלת מורכבות דומה לזו המובאת בפרק 2, תהליך העיצוב המוקדם יכול להסתכם ברשימת משימות קצרה שבהן צריכה התוכנית לטפל. אך, כשמדובר בתוכניות מורכבות יותר, תכנון טוב עשוי לכלול - חומר כתוב, גרפים, תרשימי זרימה, מסמך אבני דרך ותוכנית בקרת איכות ותיאום ציפיות. רמת התיעוד הנדרשת לפרויקט מסוים נתונה לשיקול דעתך ושל לקוחותיך (המשתמשים שעבורם מיועדת התוכנית). אך, הקפד תמיד להעלות את תכנוניך על הכתב, בצורה בהירה ומדויקת.

מכאן...

פרק זה פרש בפניך את עקרונות התכנות ב- Visual Basic. הפרקים הבאים ילוו אותך בבניית היישומים הראשונים שלך:

- **פרק 2 - יצירת התוכנית הראשונה**, מלווה אותך צעד אחר צעד ומסייע לך בבניית תוכנית Visual Basic, הראשונה שלך.
- **פרק 3 - אבני היסוד של Visual Basic**, מבאר כל אחד מן המרכיבים הבסיסיים היוצרים תוכנית Visual Basic.
- **פרק 9 - יסודות התכנות של Visual Basic**, ממשיך שלב אחד קדימה ומעמיק את יכולותיך בבניית יישומי Visual Basic.

2

יצירת התוכנית הראשונה

בפרק זה:

- יצירת ממשק המשתמש לתוכנית שלך
- קבלת אינפורמציה מהמשתמש
- שינוי מאפייני טופס
- קידוד פעולות התוכנית שלך
- הרצת התוכנית

ברוב ספרי התכנות נהוג לתת לקורא להתחיל בבניית יישום ראשוני, ובסיסי ביותר, בסגנון הדומה ל- "Hello, World" הקלאסי. בדוגמה פשוטה זו, המשתמש מתבקש להתחיל פעולה כלשהי כגון לחיצה על לחצן או הקשה על מקש, וקבלת הודעה כגון "Hello, World", כתגובה. בגירסה מעט מורכבת יותר היישום מבקש מהמשתמש להזין את שמו ומגיב בהודעה מותאמת אישית כגון "Hello, John".

למרות שדוגמאות אלו מספיקות כדי להמחיש שהינך יכול, הלכה למעשה, להשתמש בשפת התכנות כדי ליצור תוכנית, התוצאה המתקבלת אינה שימושית במיוחד. גישתנו בספר זה היא להתחיל בתוכנית דוגמה שלא רק תמחיש את העקרונות של יצירת תוכנית ב- Visual Basic, אלא גם תהיה שימושית בעולם האמיתי.

בפרק זה נבנה יישום לחישוב הלוואות שמחשב את התשלומים התקופתיים הנחוצים לכיסוי ההלוואה, בהתחשב בנתונים כגון תנאי ההלוואה ואחוז הריבית. המשתמשים שלך יוכלו להזין ולשנות ערכים אלה וערכים אחרים, לבצע את החישובים הדרושים ולקבל את תוצאותיהם. בנוסף לכך, המשתמשים יוכלו לקבל לוח זמנים של התשלומים למשך כל זמן קיום ההלוואה.

כפי שהינך רואה, הפיסקה הקודמת מתארת מה היישום יעשה, ואיזו אינפורמציה דרושה לביצוע הפעולה. כמו כן ניתן מידע לגבי הדרך שבה יש לתכנן את הממשק. זה לא היה נורא כל כך, הלא כן?

הערה:

היישום שניצור פותח מתוך תוכנה שיתופית בשם **My Amortizer**. בעזרת תכנון קפדני, אפשר להשתמש ב- Visual Basic ליצירת יישומי תוכנה ברמה מסחרית.

יצירת ממשק המשתמש לתוכנית שלך

ממשק המשתמש (User Interface) של תוכנה מתייחס לחלק, או חלקים, בתוכנית שהמשתמש רואה, ואיתם הוא מתקשר. כאשר תפתח את ממשק המשתמש של התוכנית, תוכל ללמוד כיצד להשתמש בכלים הרבים המסופקים על ידי התקן IDE (Integrated Development Environment) של Visual Basic. כפי שמוסבר בפרק 1, **להתחיל ב- Visual Basic**, התקן IDE של Visual Basic יוצר עבודה קלה בתכנון ממשק המשתמש, אשר הווה פעם משימה קשה ומסובכת במקרה הטוב.

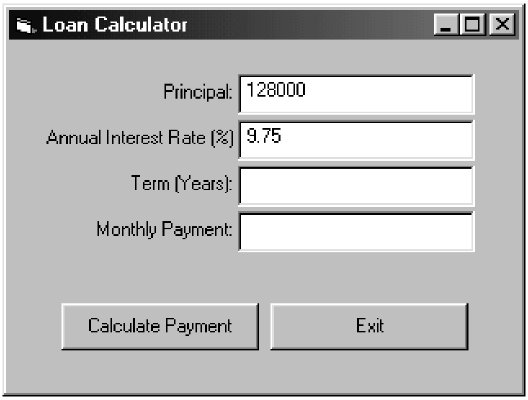
לפני שתתחיל, הבט כיצד ייראה התוצר הסופי. תרשים 2.1 מראה את המסך הראשי במחשב ההלוואות, בעודו בשימוש.

כיצד להתחיל

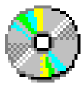
הבה נתחיל ביצירת **פרויקט** (Project) חדש. **Project** הוא בעצם סט קבצים שמכילים מידע אודות הרכיבים שיוצרים את היישום (התוכנית). כדי ליצור תוכנית Visual Basic, עליך לשנות את הרכיבים השונים באופן שיתאים ליישום שלך.

כדי להתחיל בבניית תוכנית מחשב ההלוואות, התחל בהרצת Visual Basic. מייד תראה את תיבת הדו-שיח **New Project**, כפי שמתואר בתרשים 2.2.

במידה ו-Visual Basic כבר רצה, או אם תיבת הדו-שיח New Project אינה מופיעה, בחר **File**, ואז **New Project** מתוך שורת התפריט (ראה תרשים 2.3).

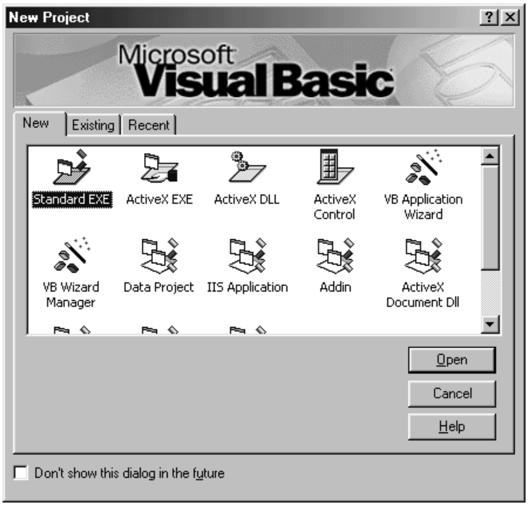


תרשים 2.1: תוכנית מחשב ההלוואות מציעה למשתמש מיגוון אפשרויות

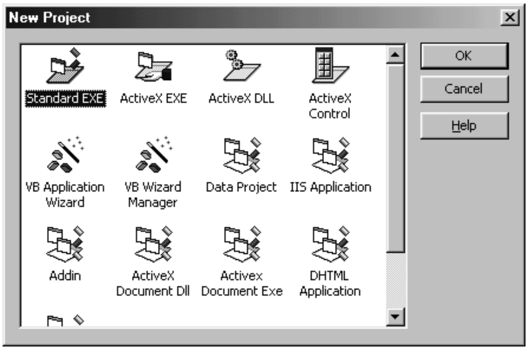


בתקליטור:

התוכנית נמצאת בתקליטור ונקראת LoanCalc



תרשים 2.2: גרסת תיבת הדו-שיח New Project המופיעה כש- Visual Basic עולה

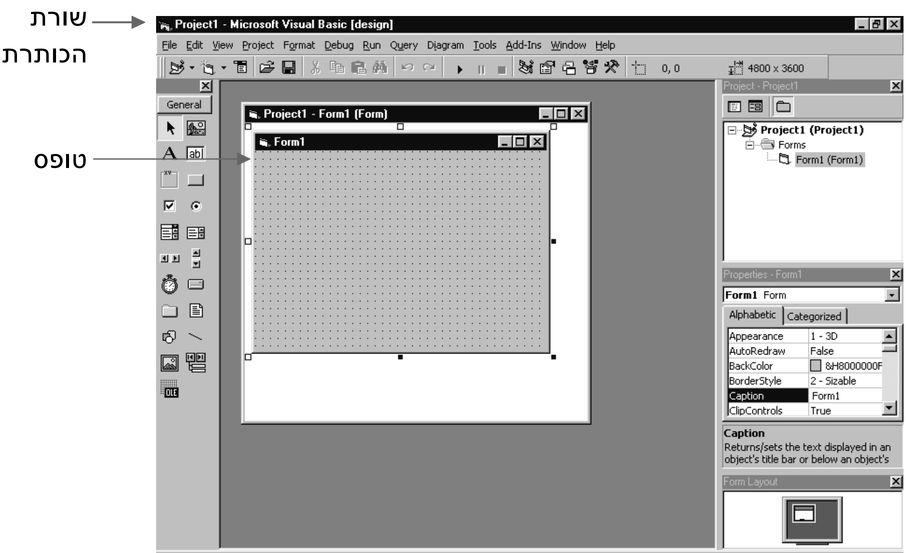


תרשים 2.3: גיריסה זו של תיבת הדו-שיח New Project מופיעה כאשר מתחילים פרויקט חדש כאשר Visual Basic כבר רצה

תיבת הדו-שיח New Project מאפשרת לנו להגדיר את סוג הפרויקט אשר ברצוננו ליצור. עבור תוכנית מחשב ההלוואות, בחר באפשרות **Standard EXE** ולחץ **OK**. מייד תועבר לסביבת הפיתוח של Visual Basic,כפי שמתואר בתרשים 2.4.

הערה:

עקב תכונות ההתאמה האישית המתקדמות של Visual Basic, המסך שתקבל עלול להיראות שונה במקצת מאשר המסך שבתרשים 2.4.

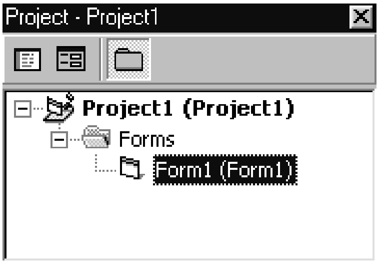


תרשים 2.4: סביבת הפיתוח של Visual Basic היא דב הציור ליצירת היישום שלך.

דבר ראשון, שים לב לשורת הכותרת של Visual Basic - במיוחד למילה Design. דבר זה אומר שהיינך נמצא כרגע ב**מצב עיצוב** (Design Mode) או **Designtime**. זהו השם שניתן לזמן בו אתם מעצב את התוכנית. מאוחר יותר, כאשר תריץ את התוכנית, Visual Basic תעבור ל**מצב ריצה** (Run Mode).

כפי שהינך רואה, תוכנית Standard EXE חדשה מורכבת מ**טופס** (Form) או חלון אחד. חלון זה יהיה, ברוב המקרים, ממשק המשתמש הראשי של התוכנית שלכם. יישומי Visual Basic מורכבים מאחד או יותר מרכיבים, כגון **טפסים** (Forms), מודולי קוד ומחלקות, וכן **פקדים** (Controls) ומרכיבים אחרים.

הבט ב- **Project Explorer** (כפי שמתואר בתרשים 2.5). הוא מורכב מרשימה של תוכן הפרויקט הנוכחי. מכיון שרק כעת התחלת פרויקט חדש, הוא מכיל טופס אחד בלבד, אשר נקרא Form1. טופס זה ממוקם בתיקיית הטפסים (Forms) של הפרויקט (הנקרא Project1 כברירת מחדל). פרויקט יכול להכיל בסופו של דבר מרכיבים רבים, ומטרת Project Explorer הוא שמירה על הסדר והארגון של הפרויקט שלך.



תרשים 2.5: Project Explorer מכיל רשימה של כל מרכיבי הפרויקט

שמירת עבודתך

בזמן כלשהו זה חייב לקרות - המערכת שלך קרסה, הפסקת חשמל, הכלב ניתק את כבל החשמל של המחשב - וכל העבודה נמחקה! כדי לצמצם את הנזק כאשר משהו מסוג זה יקרה (והוא יקרה), עליך להרגיל את עצמך לשמור את עבודתך בדיסק הקשיח מייד כאשר אתה מתחיל פרויקט חדש. לאחר מכן עליך לשמור את עבודתך בכל עת שהיא מתפתחת. עליך לכל הפחות לשמור את עבודתך לפני הרצתה. באופן זה תוכל למנוע את איבוד עבודתך אם התוכנית שכתבת תגרום למערכת לקרוס (כן, דבר זה יכול לקרות אפילו למפתחים הטובים ביותר).

שמירת פרויקטים ב-Visual Basic. תהליך שמירת הפרויקט הינו מורכב מעט יותר מכפי שאתה עלול לצפות. למרות זאת, עם מעט תרגול לא תהיה לך כל בעיה. כדי לשמור פרויקט, עליך לשמור כל אחד מהמרכיבים של הפרויקט בקובץ נפרד (יש לשמור כל טופס, מודול וכיוצא בזה). לאחר שמירת כל המרכיבים יש לשמור את הפרויקט עצמו בקובץ פרויקט. קובץ הפרויקט הוא למעשה הרשימה והמיקום של כל המרכיבים המהווים פרויקט זה.

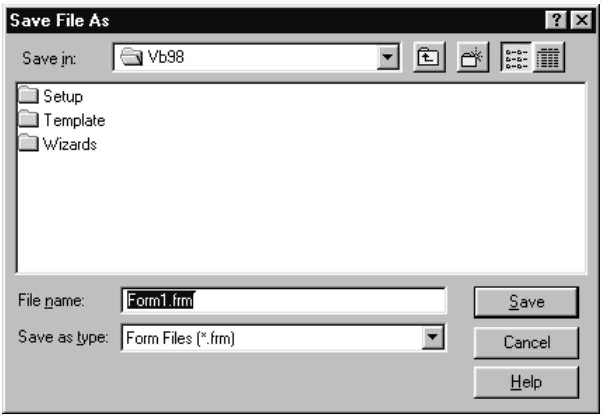
טיפ:

רצוי ליצור תיקיה נפרדת בדיסק לכל פרויקט שהינך עובד עליו. דבר זה מקל על שמירת הסדר בפרויקט, על ידי שמירת כל הקבצים השייכים לפרויקט באותו מקום. באותו אופן תוכל לשמור לאותה תיקיה כל סוג של קובץ שאינו של Visual Basic, אך קשור בפרויקט, קבצי תיעוד, גליונות אלקטרוניים, קבצי גרפיקה וכדומה.

כדי לשמור פרויקט, בחר **File**, **Save Project** מתוך התפריט של Visual Basic, או לחילופין לחץ על לחצן שמירת הפרויקט בסרגל הכלים. בפעם הראשונה שתשמור את הפרויקט (או לחילופין בכל הוספת מרכיב חדש אליו), תעבר דרך אחת או יותר מתיבות הדו-שיח של פקודת Save As (ראה תרשים 2.6), שתופיע עבור כל אחד ממרכיבי הפרויקט, וכן תיבת Save Project As עבור קובץ הפרויקט. יהיה עליך להגדיר שם ומיקום לשמירה עבור כל אחד מהקבצים שמרכיבים את הפרויקט. פקודות שמירה נוספות ישמרו את הקובץ פעם נוספת בשמו הקודם. באופן זה ניתן לבצע שמירות מהירות ותכופות על פרויקטים שמורים על ידי לחיצה על לחצן שמירת הפרויקט. במידה והתווספו מרכיבים נוספים לפרויקט מאז השמירה האחרונה, תופיע תיבת דו-שיח לשמירה עבור כל מרכיב חדש.

הערה:

Visual Basic מכילה אופציה המאפשרת לבדוק האם הפרויקט שמור כהלכה. עליך לבחור **Tools** ולאחר מכן **Options** מתוך התפריט הראשי. מתוך תפריט **Option** יש לבחור בכרטיסיה **Environment**. במסגרת **When a Program Starts**, אפשר להורות ל-Visual Basic לשמור אוטומטית כל שינוי בפרויקט ברגע שמריצים אותו, או לחילופין לתת לך להחליט האם לשמור את השינויים לפני הרצה.



תרשים 2.6: תיבת הדו-שיח Save As מאפשרת את הגדרת השם והמיקום של הקבצים המרכיבים את הפרויקט שלך

טיפ:

אפשר לשמור מרכיבים בודדים בפרויקט באופן אינדיבידואלי על ידי בחירתם ב- **Project Explorer** ובחירת **File** ולאחר מכן (component name) **Save** מן התפריט הראשי. לחילופין אפשר שמור מרכיב בודד על ידי לחיצה על הלחצן הימני של העכבר על שם המרכיב הרצוי ב- Project Explorer, ולאחר מכן בחירת האפשרות **Save** (component name) מן התפריט המופיע. אפשר לשנות את שם הקובץ על ידי שימוש באפשרות Save As בכל אחד משתי שיטות אלו.

שמירת הפרויקט מחשב ההלוואות. כעת נשמור פרויקט לדוגמה בשם "מחשב ההלוואות". בחר **File**, לאחר מכן **Save Project**, או לחילופין לחץ על לחצן שמירת הפרויקט בסרגל הכלים. בתחילה יהיה עליך לשמור את הטופס היחיד שממנו מורכב הפרויקט (Form1). בשלב זה תופיע תיבת הדו-שיח **Save As** אשר בה יוגדר שם ומיקום הקובץ.

- תיבת הטקסט **File Name** משמשת למתן שם לקובץ אשר בו יישמר מרכיב של הפרויקט. בדוגמה, השם המופיע הוא שם ברירת המחדל של הקובץ - Form1.frm. שנה שם זה לשם המתאר בדייקנות רבה יותר את תוכן הקובץ כגון LoanCalcMain. אין צורך להוסיף את הסיומת frm. לשם הקובץ. הסיומת תתווסף אוטומטית עם שמירת הקובץ.
- תיבת הרשימה הנפתחת **Save In**, אשר מופיעה בחלקה העליון של תיבת הדו-שיח, מאפשרת את הגדרת המקום אשר בו יישמר המרכיב. אם לא נשמר דבר מאז תחילת ההפעלה של Visual Basic, תוצג בפנינו תיקיית ברירת המחדל (בדרך כלל התיקיה שבה נמצאת Visual Basic). על ידי שימוש ברשימה הנפתחת מתיבה זו, ניתן לנווט אל התיקיה אשר בה ברצונך לשמור את הקובץ. במידת הצורך ניתן ליצור תיקיה חדשה לפרויקט על ידי לחיצה על לחצן **New Folder**.

אזהרה:

לעולם אל תשמור את הקבצים של מרכיבי היישום בתיקיה של קבצי התוכנה של Visual Basic. רצוי בדרך כלל ליצור ליישום תיקיה משלו.

למטרת דוגמה זו, נווט לתיקיית השורש של הדיסק הקשיח. השתמש בלחצן **Create New Folder** על מנת ליצור תיקיה חדשה. שנה את שם התיקיה החדשה מ- New Folder ל- LoanCalc. לחץ לחיצה כפולה על תיקיית LoanCalc כדי להפוך אותה לפעילה, ולחץ על **Save** כדי להשלים את הפעולה.

לאחר שמירת הטופס, תופיע תיבת הדו-שיח **Save Project As**. בתיבה זו יש להגדיר את שם ומיקום קובץ הפרויקט. מיקום ברירת המחדל של הקובץ אמור להיות המיקום שבו נשמר קובץ הטופס. שנה את שם הקובץ מ- Project1.vbp ל- Loan Calculator (הסיומת VBP. או Visual Basic Project תתווסף אוטומטית). לחץ על **Save** לסיום.

עכשיו, לאחר שבנינו את תבנית הבסיס של הפרויקט, הענקנו לו שם ושמרנו אותו, אנו מוכנים להתחיל בפיתוח האמיתי. כמובן שכדי שהתוכנית תוכל לעבד מידע, עליך לתכנן כיצד לקבל מידע זה מהמשתמש.

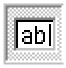
הערה:

בהתחשב באופציות שנבחרו כאשר Visual Studio הותקנה (אם Visual Basic שברשותך הותקנה כחלק ממערכת זו), עשויה להופיע תיבת דו-שיח השואלת אם ברצונך להוסיף את הפרויקט ל-SourceSafe - מערכת זו דואגת שקבצי המקור של הפרויקט יהיו מאובטחים ומגובים, ודואגת לארגון גרסאות קבצי מקור. ענה **לא** על השאלה.

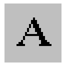
קבלת מידע מהמשתמש

מרבית תוכנות המחשב הן אינטראקטיביות. תוכנות אלו צריכות לקבל מידע מהמשתמש וכן לספק לו מידע אחר. תוכנית Visual Basic מתקשרת עם המשתמש באמצעות **פקדים** (Controls) אשר ממוקמים בתוך **טופס** (או טפסי) התוכנית. פקד הוא מרכיב אשר אנו מציבים על הטופס ואשר מתקשר עם המשתמש או עם התוכנית. רבים מהפקדים ב- Visual Basic מקבלים מידע מהמשתמש.

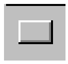
ממשק המשתמש עבור תוכנית מחשב ההלוואות יהיה אחראי על קבלת קלט, הצגת פלט, והרצת חישובי ההלוואות. אנו נשתמש בשלושה מן הפקדים הנפוצים ביותר ב- Visual Basic.



פקד תיבת הטקסט (Textbox) המשמש לקבלת מידע טקסטואלי מהמשתמש, וכן להצגת מידע מסוים חזרה למשתמש.



פקד התווית (Label) המשמש להצגת כיתוב המציג מידע למשתמש. תוויות דומות לתיבות טקסט, אך אינם ניתנות לשינוי ועריכה על ידי המשתמש.



פקד לחצן פקודה (Command Button) עליו יכול המשתמש ללחוץ כדי להפעיל פעולות של התוכנית.

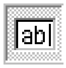
כל הפקדים הללו הינם חלק מסט הפקדים הבסיסי הנמצא בארגז הכלים של Visual Basic (ראה 2.7).



תרשים 2.7: ארגז הכלים של Visual Basic מכיל את הפקדים הנחוצים לנו לבניית יישומים

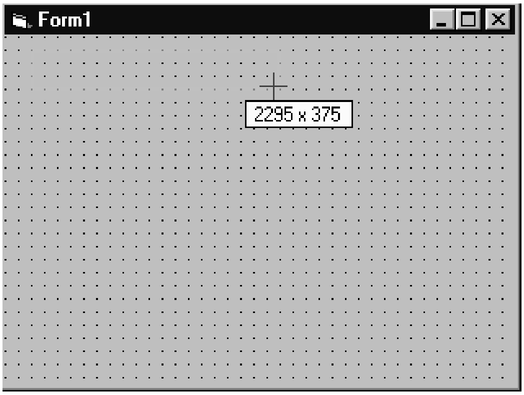
הוספת פקד תיבת הטקסט

פקד תיבת הטקסט (Text box) הינו תיבה המציגה ומקבלת טקסט. במידה מסוימת, תיבה זו דומה לתיבת טקסט שנמצאת בטפסי סקרים. התיבה יכולה לקבל קלט (מעט או עיפרון) וכן להציג מידע. פקד תיבת הטקסט הינו דוגמה להתקדמות הרבה שחלה בשפות התכנות. שפות תכנות מוקדמות יותר הצריכו מאמץ רב כדי להחליף מידע עם המשתמש. Visual Basic עובדת בשיתוף עם Windows כדי לטפל בפרטים שוליים ופשוטים כגון מיקום הטקסט על המסך, כיצד המידע יתקבל מהמשתמש וכדומה. Visual Basic צריכה להתרכז בטקסט המופיע בתוך התיבה. הפקד עצמו מטפל בשאר.



הוספת פקד לטופס. כדי ש- Visual Basic תוכל להשתמש בפקד, יש להוסיף אותו לטופס. עבור תוכנית מחשב ההלוואות נתחיל בהצבת תיבת טקסט בטופס הראשי:

1. לחץ על כלי פקד תיבת הטקסט בארגז הכלים. אם אינך בטוח איזה מהכלים הוא כלי תיבת הטקסט, השאר את מצביע העכבר מספר שניות על כל כלי, כדי לקבל **תיאור כלי** (ToolTip) על כלי זה.
2. הזז את מצביע העכבר לתוך הטופס. שים לב שהמצביע משנה צורתו לצלב. צורת הצלב מצביעה על כך שאתה עומד לצייר פקד.
3. הזז את המצביע לפינה אחת של האזור אותו ברצונך להגדיר לפקד.
4. לחץ והחזק לחוץ את לחצן העכבר.
5. גרור את העכבר לפינה הנגדית של הפקד. בעודך גורר את העכבר, שים לב שתיבת הגדרת גודל מופיעה מהפינה הראשונה של מיקום הפקד, אל המיקום הנוכחי של מצביע העכבר. תיבה זו מצביעה היכן ימוקם הפקד (ראה תרשים 2.8).



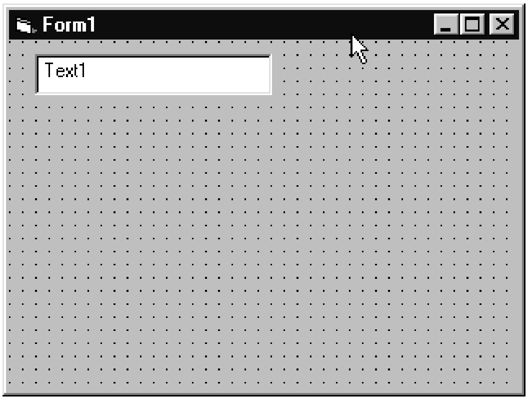
תרשים 2.8: תיבת תיאור כלי (ToolTip) מופיעה ומראה את מימדי הפקד ב-Twips (יחידת מידה מיוחדת שתתואר בהמשך הספר)

6. כאשר אתה מרוצה מגודל וצורת הפקד, שחרר את לחצן העכבר. הפקד יצויר על הטופס (ראה תרשים 2.9).



טיפ:

אפשר להוסיף פקד לטופס על ידי לחיצה כפולה על סמל הפקד בתיבת הכלים. פעולה זו ממקמת את הפקד בגודל ברירת מחדל במרכז הטופס. לאחר מכן אפשר להזיז ולשנות את גודלו לגודל הרצוי בעזרת נקודות האחיזה, כפי שמתואר בקטע הבא, **הזזה ושינוי גודל הפקד**.



תרשים 2.9: תיבת הטקסט המוכנה מופיעה על הטופס. לאחר הופעתה תיבה זו מסומנת כנבחרת (שימו לב לנקודות האחיזה לשינוי הגודל המופיעות מסביב לפקד)

הדרך המשמשת להצבת תיבת הטקסט זהה לזו המשמשת להצבת רוב הפקדים בטופס.

קביעת מאפייני הפקד. לאחר שהוספנו פקד לטופס, בדרך כלל נרצה לשנות אחד או יותר ממאפייניו. מאפיינים הינם הגדרות ששולטות במראה ובהתנהגות האובייקט. לתיבת הטקסט שהוספנו, נרצה לקבוע את המאפיינים **Name** (שם) ו-**Text** (טקסט).

השם (Name) במאפיינים הוא חשוב מאוד. הוא משמש בקוד התוכנית על מנת לזהות את הפקד. מכיון שלתוכנית עלולים להיות פקדים רבים מאותו סוג, אנו יכולים להשתמש במאפיין השם בכדי לזהות פקד מסוים שעבורו נכתב קטע קוד כלשהו.

לכל פקד חייב להיות שם, אשר מיוצג על ידי ערך המאפיין Name שלו. בנוסף לכך, לכל פקד בטופס מסוים חייב להיות שם ייחודי, אלא אם הוא חלק ממערך פקדים (מערכי פקדים מוסברים לפרטים בפרק 13 **עבודה עם מערכי פקדים**. לעת עתה עלינו לוודא שכל הפקדים יקבלו שם ייחודי).

Visual Basic מציבה שם ברירת מחדל לכל פקד הממוקם בטופס. מכיון שתיבה זו היא התיבה הראשונה שהצבנו בטופס, שמה יהיה Text1. תיבות נוספות תיקראנה Text2, Text3 וכדומה. רצוי מאוד לשנות את שם ברירת המחדל לשם בעל יותר משמעות. הנח לדוגמה, שיש לך שלוש תיבות טקסט בטופס, שמשמשות לקבלת פרטים מהמשתמש: שם משפחה, שם פרטי וכתובת. אם תשנה את שמות ברירת המחדל של Text1, Text2, Text3 לשם בעל משמעות, כמו txtLname, txtFname, txtAddress, יהיה קל הרבה יותר לזכור עבור מה משמשת כל תיבה בזמן מאוחר יותר.

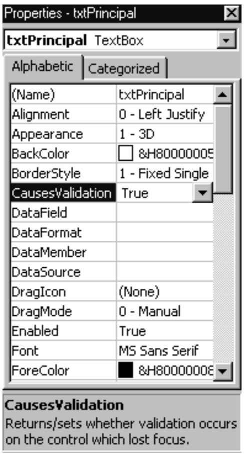
טיפ:

שים לב ששמות הפקדים שהוצעו כאן מתחילים בקידומת txt. זהו נוהג נפוץ להתחיל שם פקד בקידומת בעלת שלוש אותיות המתארת את סוג הפקד. כך, בעת איתור באגים, ברור מייד ש-txtLname מתייחס לפקד תיבת טקסט שמכיל מידע על שם משפחה. קידומות נפוצות אחרות הם lbl לתוויות ו- cmd לפקד לחצן פקודה.

כדי לשנות את השם של תיבת הטקסט הראשונה שהצבנו על הטופס, עליך לוודא שהפקד הוא נבחר. הפקד הנבחר הוא הפקד אשר את המאפיינים שלו ניתן לשנות בחלון המאפיינים. כפי שנראה בהמשך ניתן לבחור מספר פקדים יחדיו. ניתן לדעת אם פקד הוא הפקד הנבחר על ידי כך שמופיעים מסביבו סדרת נקודות אחיזה לשינוי גודלו. שם הפקד הנבחר מופיע בתיבת האובייקט של חלון המאפיינים. תיבה זו הינה רשימה נפתחת המופיעה מתחת לכותרת חלון המאפיינים. אם הפקד הרצוי אינו נבחר, עליך פשוט ללחוץ עליו בכדי לבחור אותו.

עכשיו כאשר בחרת את הפקד, הבט בחלון המאפיינים. העמודה השמאלית בחלון המאפיינים הנה רשימת מאפיינים התקפים לגבי האובייקט הנבחר בזמן התכנון. הערך הנוכחי של כל מאפיין מופיע בצד הימני של חלון המאפיינים. חפש את המאפיין name בעמודה השמאלית, בתחילת הרשימה (מכיון שמאפיין השם הוא חשוב כל כך, הוא מופיע ראשון ברשימה. המאפיינים האחרים מסודרים בסדר אלפביתי. יש אפשרות לראות את סדר המאפיינים לפי קטגוריה על ידי בחירת הכרטיסיה **Categorized**. לחץ על המאפיין name והפוך אותו למאפיין הנוכחי (שים לב ששם המאפיין מודגש). שים לב לשם ברירת המחדל - Text1 המופיע בצד הימני של המאפיין. בשלב זה, כל מה שעליך לעשות הוא להקליד שם חדש עבור המאפיין name או לערוך אותו. שנה את שם מאפיין name בפקד תיבת הטקסט על ידי הקלדת שם בעלת משמעות - הבה נשתמש בשם txtPrincipal. לאחר ההקלדה הקש Enter. תרשים 2.10 מראה את חלון המאפיינים לאחר שינוי זה.

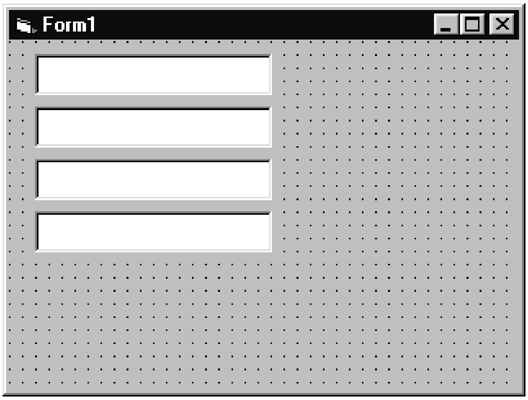
המאפיין text בפקד תיבת הטקסט מייצג את הטקסט המוצג בתוך התיבה כאשר היא תופיע על המסך. כאשר המשתמש מקליד טקסט בתוך התיבה, Visual Basic משנה את ערך המאפיין text כך שיתאים לערך שבתיבה. כברירת מחדל, תיבת טקסט חדשה מכילה את שמה. בתיבת הטקסט החדשה שיצרת מופיע השם Text1 במאפיין text שלו (זכור ש-Text1 היה שם ברירת המחדל של הפקד כאשר יצרת אותו). במקרה שלנו אין אנו רוצים דבר בתוך התיבה כאשר היא תופיע על המסך. לשם כך עליך לבחור את מאפיין text בחלון המאפיינים. בחר את הערך הנוכחי (Text1) ומחק אותו.



תרשים 2.10: השתמש בחלון המאפיינים כדי לשנות את ערכי הפקדים

הוספת שאר תיבות הטקסט. כעת, כאשר הוספת תיבת טקסט אחת לטופס, לא צריכה להיות בעיה להוסיף את השאר. הוסף שלוש תיבות טקסט נוספות לטופס. קרא להן txtIntRate, txtTerm ו-txtPayment. בנוסף לכך מחק את תוכן המאפיין text שלהן. לכשתסיים, הטופס שלך אמור להיראות דומה לטופס בתרשים 2.11.

אם הטופס שלך לא נראה בדיוק כמו הטופס בתרשים 2.11, אל תדאג. אחד הדברים הנחמדים בסביבת תכנון ויזואלית, הוא היכולת לשנות את מראה האובייקטים בקלות רבה.



תרשים 2.11: יש ארבע תיבות טקסט בממשק המשתמש של מחשב ההלוואות



טיפ:

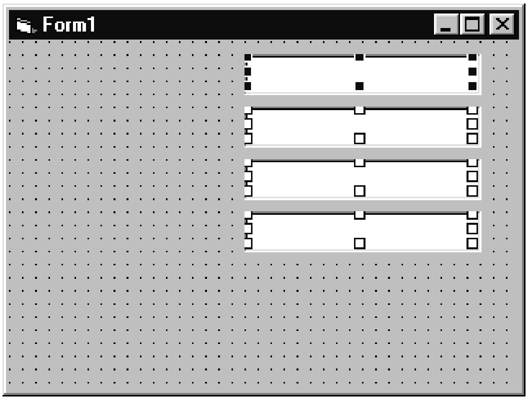
כדי לצייר מספר פקדים בעת ובעונה אחת, הקש ctrl בזמן שאתה בוחר כלי פקד מתוך תיבת הכלים. כלי זה ימשיך להיות הכלי הנבחר גם לאחר שסיימת לצייר את הפקד. תוכל להמשיך לצייר פקדים זהים מבלי שתצטרך לבחור את הכלי מחדש ברגע שסיימת לצייר, בחר את כלי הסמן של ארגז הכלים כדי לחזור למצב הרגיל.

הזזה ושינוי גודל הפקדים. אם אינך מרוצה מהמיקום שבו ממוקם הפקד, השתמש בעכבר כדי לגרור אותו למיקום החדש. שים לב שאם אתה עוצר במהלך הגרירה למספר שניות, המיקום היחסי של הפקד יופיע כתיאור כלי (ToolTip).

אם ברצונך לשנות את גודל הפקד, עליך לבחור אותו - דבר שיגרום לנקודות האחיזה להופיע מסביב לו. לאחר הופעת הנקודות אפשר לגרור אותן בעזרת העכבר בכדי לשנות את גודל הפקד. הנקודות בקצה העליון והתחתון של הפקד שולטות בגובה שלו. הנקודות בקצה ימני והשמאלי שולטות ברוחב שלו. הנקודות שבקצוות משנות גם את הגובה וגם את הרוחב בעת ובעונה אחת. שים לב כיצד תיבת שינוי הגודל מופיעה בעוד אתה משנה את גודל הפקד, זאת כדי להמחיש את גודלו הסופי.

עכשיו כאשר יצרת את תיבות הטקסט המשמשות לקלט מהמשתמש, עליך לסמנן בתוויות כדי שהמשתמש יידע מה להקליד בכל תיבה. כדי לפנות מקום לתוויות, הזז את תיבות הטקסט לצד ימין. תוכל לגרור כל תיבה בנפרד, אך הדרך המהירה יותר היא לגרור את כולן בבת אחת. כדי לבחור את כל קבוצת התיבות, כל שעליך לעשות

הוא לבחור תיבה אחת, להקיש ctrl ולהשאיר אותו לחוץ, ולבחור את כל שאר התיבות. שים לב שכל התיבות שבחרת באופן זה מקבלות נקודות אחיזה משלהן. לאחר שבחרת את כולן, גרור אחת מהן. בעוד אתה גורר, כל הקבוצה נגררת בו-זמנית. גרור את התיבות לצד ימין של הטופס. לכשתסיים, הטופס שלך אמור להיראות דומה לתרשים 2.12.



תרשים 2.12: בחירת מספר פקדים בו-זמנית מאפשרת את הזזתם ביחד כקבוצה



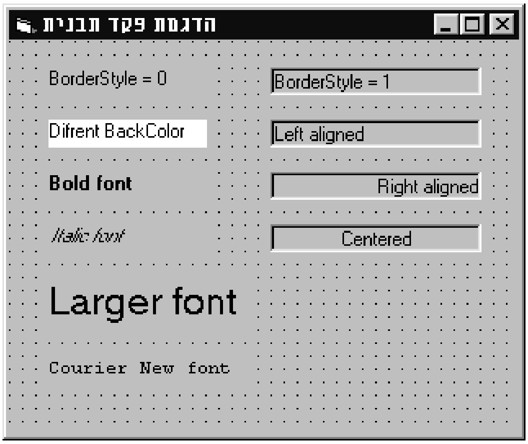
טיפ:

במקום לבחור קבוצה על ידי לחיצה על ctrl תוך כדי בחירת הפקדים, אפשר לסמן את כל הקבוצה על ידי ציור מלבן מסביבם. לחץ והחזק את הלחצן השמאלי של העכבר לחוץ באזור ריק בטופס. שים לב שנמתח מלבן בעוד אתה מזיז את העכבר. צייר תיבה מסביב לכל הפקדים שברצונך לבחור.

יצירת תוויות עבור פקדי התוכנית

מובן מאליו שהמשתמש חייב לדעת איזה ערכים עליו להקליד בכל אחת מתיבות הטקסט. הדרך הקלה ביותר לבצע זאת היא להוסיף פקדי תווית ליד כל אחת מתיבות הטקסט. התווית תשמש ככותרת עבור תיבת הטקסט, ותכיל תיאור קצר של סוג המידע שיש להקליד לתוכה.

למרות שעבור המשתמש יש הבדל גדול בין תיבות טקסט ותוויות, הן למעשה דומות מאוד מבחינת המפתח. שני סוגי הפקדים יכולים להכיל אותו סוג טקסט, רק שתווית אינה ניתנת לעריכה על ידי המשתמש. למרות זאת, על ידי שינוי מאפיינים שונים של פקד התווית, מראה הטקסט שבתוכה יכול להשתנות בדרכים רבות. תרשים 2.13 ממחיש סוגי מראות שונים של פקד התווית.

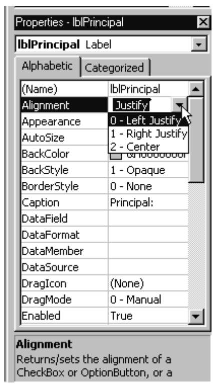


תרשים 2.13: תוויות יכולות לקבל גדלים שונים וצורות הצגה רבות ושונות

אם נמשיך את ההשוואה עם טופס סקרים, אזי פקדי תוויות הם המקבילה למילים המודפסות על הטופס כדי לתאר את תיבות הטקסט הריקות. ההבדל המרכזי בין תווית ותיבת טקסט הוא שתווית מכילה טקסט שהמשתמש אינו יכול לשנות. הטקסט הנכלל בפקד תווית נשמר במאפיין Caption (בניגוד לתיבת טקסט, שם נשמר מידע זה במאפיין Text).

כדי להוסיף תווית לטופס, בצע את הפעולות הבאות

1. בחר את כלי פקד התווית מתוך ארגז הכלים.
2. צייר את פקד התווית משמאל לתיבת הטקסט הראשונה.
3. שנה את המאפיין Name של התווית ל- lblPrincipal בחלון המאפיינים.
4. שנה את המאפיין Caption ל- :Principal.
5. שנה את המאפיין Alignment ל- 1-Right Justify, על ידי שימוש בחץ המופיע ליד הגדרות המאפיין (ראה תרשים 2.14). פעולה זו מיישרת את הכיתוב לימין, ליד תיבת הטקסט התואמת.

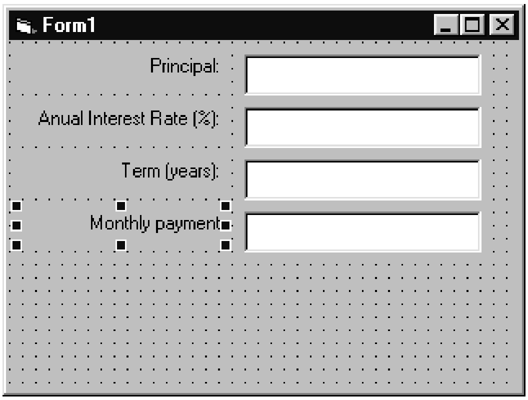


תרשים 2.14: לחיצה על החץ הנפתח במאפיין Alignment מאפשר בחירה מתוך רשימת אפשרויות קיימות

בשלב זה הינך אמור לראות את כיצד פקד התווית עוזר למשתמש להבין מה להקליד בתיבת הטקסט הראשונה. צור עוד שלושה פקדי תווית, אחד עבור כל תיבת טקסט שנותרה. השתמש בערכים המומלצים בטבלה 2.1 עבור מאפייני Name ו-Caption (אל תשכח לשנות גם את המאפיין Alignment). לכשתסיים, הטופס שלך אמור להיראות דומה לתרשים 2.15.

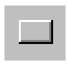
טבלה 2.1: מאפייני Name ו-Caption עבור פקדי התווית של תוכנית מחשב ההלוואות

מאפיין Name	מאפיין Caption
lblPrincipal	Principal:
lblIntRate	Annual Interest Rate (%):
lblTerm	Term (years):
lblPayment	Monthly payment



תרשים 2.15: תוויות התוכנית מציגות למשתמש מה להקליד בתיבות הטקסט

הוספת לחצני פקודה



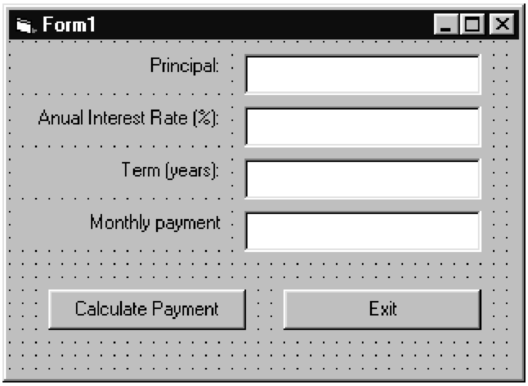
עד כה, יישום הדוגמה שלך מכיל תיבות טקסט לקבלת קלט מהמשתמש, וסדרת תוויות המתארות תיבות אלו. עליך ליצור דרך שבה המשתמש יוכל להתחיל פעולות - לשם כך נועד פקד **לחצן פקודה** (CommandButton). המשתמש יכול ללחוץ על לחצן זה כדי לגרום למשהו לקרות. תוכל להוסיף לחצן פקודה לטופס באותו אופן שבו הוספת כל פקד אחר - על ידי שימוש בעכבר כדי לצייר אותו.

לחצני פקודה, בדומה לתוויות, הינם בעלי מאפיין Caption המאפשר לקבוע איזה טקסט יופיע בתוך הלחצן, כך שהמשתמש יידע למה הוא משמש.

כדי להשלים את ממשק המשתמש של תוכנית מחשב ההלוואות, הוסף שני לחצני פקודה בתחתית הטופס. שנה את מאפייני Name ו-Caption על פי טבלה 2.2. תרשים 2.16 מראה כיצד הממשק השלם אמור להיראות.

טבלה 2.2: מאפייני Name ו-Caption עבור לחצני הפקודה של תוכנית מחשב ההלוואות

מאפיין Name	מאפיין Caption
cmdCalculate	Calculate Payment
cmdExit	Exit



תרשים 2.16: לאחר הוספת לחצני הפקודה, סיימנו את ממשק המשתמש של התוכנית

שינוי מאפייני הטופס

הינך יכול לשנות את מאפייני הטופס באותו האופן שבו אתה מגדיר את המראה והתנהגות הפקד.

בדיוק כמו בפקדים, לטפסים יש מאפיין Name. שם ברירת המחדל עבור הטופס הראשון בפרויקט הינו Form1. טפסים נוספים יקבלו שמות עוקבים כגון Form2, Form3, Form4 וכדומה. כמו בפקדים, כדאי מאוד לתת לטופס שם כעל משמעות. השתמש בחלון המאפיינים כדי לשנות את המאפיין Name עבור הטופס היחיד של פרויקט מחשב ההלוואות, ל-frmMain.

כדי לשנות את רוחב הטופס, בצע את הפעולות הבאות :

1. לחץ על אזור ריק בטופס כדי לבטל בחירת פקד שאולי נבחר. הפעולה מהווה למעשה בחירה של הטופס. שים לב ששם הטופס מופיע בחלון המאפיינים בתיבת האובייקטים.
2. עכשיו כשהטופס נבחר, המאפיינים המוצגים בחלון המאפיינים, מהווים למעשה את מאפייני הטופס. גלול את חלון המאפיינים עד אשר תמצאו את המאפיין Width, לקראת סוף הרשימה, ושים לב לערך המופיע בו.
3. שנה את רוחב הטופס על ידי לחיצה עם העכבר על נקודת האחיזה הימנית וגרירתה (אם שינית את ההגדרות כך ש- Visual Basic תרוץ במצב SDI - הסבר בהמשך הספר - עליך רק לגרור את קצה החלון כדי לשנות את גודלו).

4. כדי לשנות את המאפיין Width חזרה לערכו המקורי, בחר את הערך החדש שמופיע בחלון המאפיינים ושנה אותו לערך הקודם. ברגע שתקיש Enter, הטופס ישנה את גודלו בחזרה לגודל המקורי.

שמירת עבודתך - שוב...

לאחר שסיימת לבנות את ממשק המשתמש לתוכנית מחשב ההלוואות, יהיה זה זמן מצוין לשמור שוב את עבודתך. מכיון שכבר שמרת את הפרויקט, שמירה חוזרת היא פעולה פשוטה ביותר - ומומלצת ביותר. למעשה עדיף לשמור את העבודה במספר שלבים במשך תהליך הפיתוח. לעולם אין לדעת מתי תתעוררנה בעיות. שמירה מהירה לוקחת מספר שניות בלבד, ועשויה לחסוך עבודה רבה.

כדי לשמור שוב את הפרויקט, יש לבחור **File** ואז **Save Project** מהתפריט הראשי. או לחילופין יש ללחוץ על לחצן שמירת הפרויקט מתוך סרגל הכלים. מכיון שכבר אמרת ל- Visual Basic היכן לשמור את הקבצים המהווים את הפרויקט, הם יישמרו אוטומטית באותו מקום, ובאותו שם קובץ. במידה והתווספו רכיבים חדשים (טפסים, מודולים וכדומה) מאז השמירה האחרונה, יהיה עליך להגדיר את שם הקובץ החדש וכן את המיקום בו יש לשמור אותו.

קידוד פעולות התוכנית שלך

כפי שהוזכר קודם לכן, ממשק המשתמש של תוכנית מחשב ההלוואות כבר מוכן. למרות זאת, התוכנית עדיין אינה עושה דבר בשלב זה. כדי להפוך את התוכנית לפונקציונלית, יהיה עליך להוסיף לה קוד. המונח קוד, כפי שהוא מופיע בספר, מתייחס לשורה אחת או יותר של פקודות תכנות, הכתובות בשפת תכנות כלשהי (Visual Basic במקרה שלנו).

תגובה לאירועים

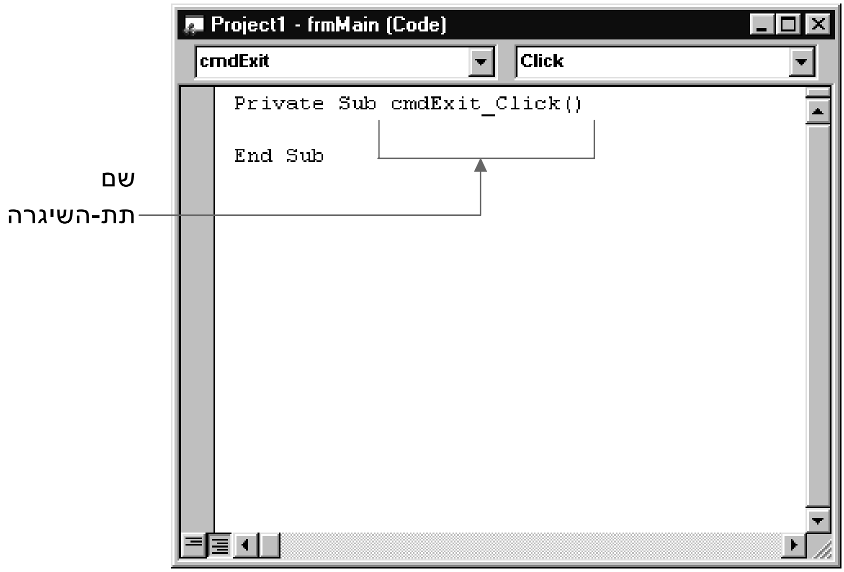
Visual Basic היא תוכנה **מונחית אובייקטים** (Object-Oriented), **ומונעת אירועים** (Event-Driven). משמעותו של דבר זה היא שממשק המשתמש של התוכנית מורכב מאובייקטים (פקדים, טפסים וכיוצא בזה). יש "ללמד" את התוכנית איזו פעולה לבצע כאשר אירועים קורים לאובייקטים אלה.

אירוע הוא בדרך כלל פעולה אשר המשתמש יזם. יש ללמוד לצפות אילו אירועים עשויים (ואמורים) להשפיע על האובייקטים בתוכנית, ולכתוב את הקוד המתאים לטיפול באירועים אלה. לדוגמה, במקרה של לחצן הפקודה Exit, הקוד צריך להגיב לאירוע לחיצה - Click, על הלחצן, ביציאה מן התוכנית. קטע קוד זה צריך להיות מופעל בכל פעם שאירוע Click של הלחצן Exit מופעל.

כדי לגרום לתוכנית להגיב לאירועים עליך להציב קוד עם **שגרות אירועים** (Event Procedures). שגרת אירוע היא קטע קוד שמבוצע ברגע שאירוע מסוים מתרחש לאובייקט מסוים.

כדי להגיב למקרה שבו המשתמש ילחץ על לחצן Exit, עליך להוסיף קטע קוד לשגרת האירוע Click של הלחצן. הבה נמחיש זאת על ידי כתיבת קוד עבור שגרת האירוע Click של לחצן Exit של תוכנית מחשב ההלוואות.

לחץ לחיצה כפולה על הלחצן Exit שהצבת על הטופס של יישום הדוגמה. פעולה זו תפתח חלון חדש הנקרא חלון הקוד (ראה תרשים 2.17). אפשר לפתוח חלון קוד נפרד עבור כל טופס (או כל מודול אחר) שמופיע בפרויקט. בחלון זה ממקמים את הקוד שמתייחס לטופס ולאובייקטים שבו. שים לב שחלון הקוד מכיל תבנית, או מעטפת, של תת-שיגרה, ואשר מתחילה במילים Private Sub ומסתיימת במילים End Sub. תת-שיגרה (Sub procedure) או בפשטות שיגרה, הוא רצף קוד בעל שם ואשר מבוצע כיחידה אחת.



תרשים 2.17: חלון הקוד הוא תוכנת עריכה רבת תכונות שמתקשרת לאובייקטים שבתוכנית

החלק המופיע אחרי המילים Private Sub מציין את שם תת-השיגרה. תת-שיגרה זו נקראת cmdExit_Click, שם אשר נקבע מראש ובא לציין את שגרת האירוע Click עבור הפקד הנקרא cmdExit. Visual Basic תבצע כל קטע קוד הממוקם בתוך תת-שיגרה זו, כל פעם שאירוע Click מתרחש על לחצן פעולה זה.

הערה:

רוב הפקדים יכולים להגיב למספר אירועים שונים. לכל סוג פקד יש אירוע ברירת מחדל. אירוע זה הינו האירוע הנפוץ ביותר לסוג זה של פקד. לדוגמה, אירוע Click הוא האירוע הנפוץ ביותר לפקד לחצן פעולה. אירוע ברירת המחדל הוא האירוע שנפתח בחלון הקוד כאשר לוחצים לחיצה כפולה על פקד חדש.

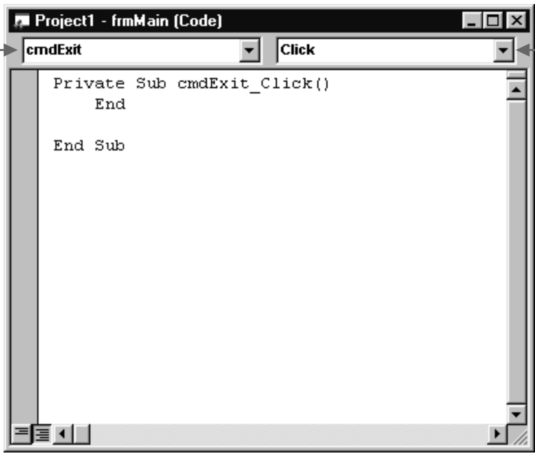
כדי לגרום לתוכנית להפסיק ברגע שהמשתמש לוחץ על הלחצן Exit, כל שעליך להוסיף היא שורת קוד בודדת לשיגרה בין Private Sub cmdExit_Click() לבין End Sub. הסמן אמור להיות ממוקם בין פקודות אלו. הקש TAB כדי ליצור כניסה לשורת הקוד (כדאי להתחיל את הקוד מעט קדימה כדי לשפר את קריאותו) והקלד את המילה End. הקש Enter כדי להוסיף שורת רווח (אין צורך לעשות זאת אך הדבר משפר את קריאות הקוד). לאחר שתסיים, הקוד השלם אמור להיראות כך:

```
Private Sub cmdExit_Click()
   End
End Sub
```

ברכותינו! כתבת כרגע את הקוד הראשון שלך ב- Visual Basic.

הגדרת שגרות אירוע

הבט בשתי תיבות הרשימה הנפתחת שנמצאות בקצהו העליון של חלון הקוד, כפי שמוצג בתרשים 2.18. תיבת האובייקט (התיבה השמאלית) נותנת את רשימת כל האובייקטים שמוקמו בטופס הנוכחי, וכן את הטופס עצמו. תיבת השיגרה (מצד ימין) נותנת את רשימת כל האירועים שתקפים לגבי סוג האובייקט שנמצא כרגע בחלון האובייקט. בעזרת שתי תיבות הרשימה הנפתחת האלו ניתן לנווט לכל מקום בחלון הקוד. חשוב על קוד עבור טופס מסוים כעל קובץ טקסט ארוך. חלון הקוד הוא כלי ניווט שעוזר לנו להציג במהירות כל שילוב בין אובייקט לאירוע.



תרשים 2.18: שגרת האירוע Click של הלחצן Exit תגרום לתוכנית להפסיק

חלון הקוד יופיע אוטומטית ברגע שתלחץ לחיצה כפולה על פקד כלשהו בעת התכנון. החלון ייפתח באירוע ברירת המחדל עבור הפקד שנבחר, אלא אם כן יש אירוע אחר המופיע כבר בקוד הפקד. במקרה זה האירוע המופיע בקוד יוצג למקרה שנצטרך לערוך את הקוד. תוכל כמובן, להשתמש בתיבות האובייקט והשיגרה בכל עת כדי למצוא במהירות כל אובייקט או אירוע מבוקש.

טיפ:

בנוסף ללחיצה כפולה על הפקד, אפשר לפתוח את חלון הקוד בהקשת F7, לחיצה על לחצן **View Code**, או על ידי בחירת **View** ואז **Code**.

כעת כאשר הקוד ללחצן Exit מוכן, כל מה שנשאר הוא לקדד את לחצן Calculate Payment. לחצן זה נועד לחשב את התשלום החודשי לכיסוי ההלוואה, בהתחשב בנתונים שהוכנסו על ידי המשתמש. קוד זה ייכתב כשגרת אירוע Click עבור לחצן הפקודה cmdCalculate. ניתן להציג את שגרת האירוע Click של הלחצן cmdCalculate על ידי הבאת עיצוב הטופס לקדמת המסך ולחיצה כפולה על cmdCalculate. לעומת זאת, מכיון שחלון הקוד פתוח ממילא, יהיה זה יעיל יותר לבחור את cmdCalculate מתוך תיבת הגלילה הנפתחת של האובייקט. דבר זה מציג את השיגרה cmdCalculate_Click בחלון הקוד.

כתיבת קוד בתוכנית

השיגרה אשר מחשבת את חישובי ההלוואות תהיה מסובכת יותר משגרת היציאה מהתוכנית. מובן שהקוד שמבצע את החישובים יהיה מורכב יותר מאשר הצהרת End פשוטה. בנוסף, שיגרה זו תצריך עבודה נוספת, מכיון שנשתמש בה במשתנים.

משתנה (Variable) הינו מאגר מידע זמני. במקרים רבים התוכנית שתבנו תצטרך לזכור מידע כמו תוצאות חישובים, שם המשתמש, סיכומי הזמנות ועוד, במהלך ריצתה. חשוב על משתנים כלוח שהתוכנית יכולה להשתמש בו כדי לזכור מידע. התוכנית, במסגרת קווים מנחים מסוימים, יכולה לרשום מידע בלוח לשנותו ואף למחוק אותו לחלוטין.

ראה: היכרות עם משתנים בפרק 8.

הצהרה על משתנים. החלק הראשון של שגרת החישוב ישמש להגדרת המשתנים שאותם נצטרך. דבר זה אומר שנאמר ל- Visual Basic את שמות המשתנים שישמשו את הפונקציה, וכן נגדיר איזה סוג מידע יכיל כל משתנה. למרות ש- Visual Basic לא מחייבת הגדרת משתנים (כברירת מחדל), רצוי לעשות זאת בכל מקרה.

טיפ:

הגדר תמיד את המשתנים שלך! ודא שהאופציה **Require Variable Declaration** בתפריט **Tools**, **Options** בכרטיסיה **Editor**, מסומנת כפעילה. דבר זה גורם ל- Visual Basic לדווח על שגיאות אם אתה מנסה להריץ תוכנית שמשתמשת במשתנה שלא הוגדר (לדוגמה אם טעית באיות שם המשתנה).

השיגרה Calculate משתמשת בארבעה משתנים. משתנה אחד עבור כל ערך של כלל ההלוואה, אחוז הריבית, תנאי ההלוואה, והחישוב החודשי. ודא שהסמן נמצא בשורה הריקה שבין Private Sub cmdCalculate_Click() לבין End Sub. הקש Tab כדי ליצור כניסה לשורת הקוד, והקלד את שורת הקוד הבאה:

```
Dim cPrincipal As Currency
```

שים לב שכאשר מקישים על Enter לאחר כתיבת הקוד, הסמן לא חוזר לקצה השמאלי של חלון הקוד אלא נשאר בכניסה שהגדרת בעזרת Tab. אפשר להגדיל את הכניסה על ידי לחיצה על Tab, ולהקטין אותה על ידי Backspace.

כפי שתראה מהקוד, הפורמט הכללי להגדרת משתנה היא המילה Dim, אחריה שם המשתנה, המילה As וסוג המשתנה. כמו באובייקטים, יש ליצור סטנדרט לשמות. לכל אורך הספר, תראה סטנדרט נפוץ מאוד לסימון משתנים אשר משתמש באות הראשונה של שם המשתנה כקידומת המגדירה את סוגו. שארית השם הוא תיאור מטרת המשתנה. לדוגמה, המשתנה cPrincipal, הוא מסוג Currency ומכיל את סך כל ההלוואה. חלק מסוגי המשתנים והקידומות שלהם מוצגים בטבלה 2.3.

טבלה 2.3: קידומות המשמשות למתן שם למשתנים על פי הסטנדרט הנפוץ

קידומת	סוג המשתנה
s	מחרוזת (String)
n	מספר שלם (Integer)
l	מספר שלם ארוך (Long Integer)
f	נקודה צפה עם דיוק יחיד (Single-precision Floating Point)
d	נקודה צפה עם דיוק כפול (Double-precision Floating Point)
c	מטבע (Currency)
b	בוליאני (אמת/שקר) (Boolean)
v	משתנה מסוג Variant

הוסף את שורת הקוד הזאת להצהרה הבאה:

```
Dim fIntRate As Single
```

מספר משתנים יכולים להיות מוגדרים על ידי אותה שורת קוד. אם תעשה כך, יש לרשום בנפרד את כל סוגי המשתנים:

```
Dim nTerm As Integer, cPayment As Currency
```



אזהרה:

טעות נפוצה למתחילים היא להגדיר מספר משתנים בהצהרה כמו Dim x, y, z As Integer. עליך לזכור שיש להגדיר לכל משתנה את הסוג שלו באופן מפורש. במקרה הנזכר רק z יוגדר כמשתנה מסוג Integer ו-x ו-y יוגדרו כמשתנים מסוג Variant. סוג זה של משתנה הוא סוג מיוחד אשר יוסבר בהמשך הספר. הכתיבה הנכונה לקטע מסוג חייבת להיות כתיבת As Integer אחרי כל אחד משמות המשתנים.

קוד השיגרה. קוד השיגרה עצמו הוא החלק שלמעשה מבצע את העבודה. בתוכנית מחשב ההלוואות שלנו, חלק זה של השיגרה יהיה אחראי על אחזור ערכי הקלט משלוש תיבות הטקסט הראשונות, חישוב התשלום החודשי, והצגתו בתיבת הטקסט הרביעית.

הקלד שתי שורות אלו לחלון הקוד (לאחר הגדרת המשתנים):

```
'Store the principal in the variable cPrincipal
cPrincipal = Val(txtPrincipal.Text)
```

השורה הראשונה היא הערה שנועדה להסביר מה מתרחש בקטע זה. הערה מסומנת בגרש (') בהתחלת השורה, בדרך כלל. ברגע ש- Visual Basic נתקלת בגרש, היא מתעלמת משאר השורה. Visual Basic עוברת לשורה הבאה להמשך הפעולות.

השורה השנייה בקוד מאחזרת את המידע שהמשתמש הכניס בתיבת הטקסט txtPrincipal, ומאחסנת את הערך שלו במשתנה cPrincipal. פעולה זו מבוצעת בעזרת הצהרת השמה, בדומה לאלגברה, כאשר המשתנה מופיע בצד שמאל של סימן שווה והערך מופיע בצד ימין. הפונקציה ()Val משמשת להמרת כל מה שמופיע בסוגריים של הפונקציה לערך מספרי עבור החישובים. הערך הממשי של תיבת הטקסט מאוחזר על ידי גישה למאפיין Text (מתייחסים למאפיינים של אובייקט בתוך הקוד על ידי שימוש בפורמט Object.Property. חשוב על txtPrincipal.Text כעל הערך הנוכחי של המאפיין של txtPrincipal (זכור שהמאפיין Text של תיבת הטקסט מכיל את הערך הנוכחי שלו על המסך).



טיפ:

בעודך מקליד את שאר הקוד, שים לב להערות הנוספות המופיעות בגוף הקוד. להזכירך, הערה היא שורת קוד שאינה מבוצעת. היא נועדה להסביר את משמעות הקוד עצמו. ההערות הינן סוג של תיעוד העוזר למפתח להבין בקלות את משמעותו של קטע הקוד שהוא צריך לערוך. זהו רעיון טוב מאוד, ברוב המקרים, להוסיף הערות רבות לקוד. אתה עלול לחשוב שתזכור מדוע פתרת בעיה בדרך כלשהי, אך כשתביט בקוד שכתבת לפני שנה (או מישהו אחר), ההערות תהפוכנה את הקוד למובן הרבה יותר.

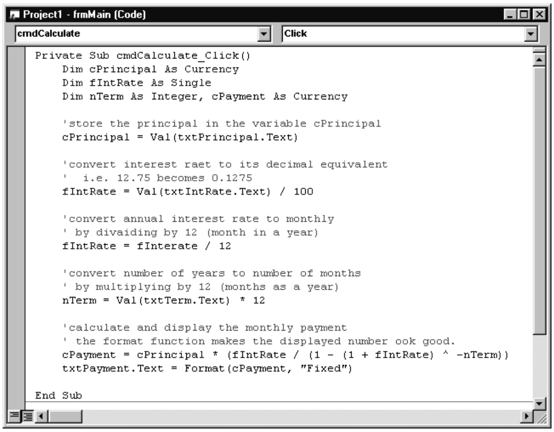
הקלד בחלון הקוד את שארית הקוד כפי שמוצג ברשימה 2.1. ההערות אמורות לעזור לך להבין כיצד קוד זה עובד. תרשים 2.19 מראה את תת-השיגרה השלמה כפי שהיא מופיעה בחלון הקוד.

```
'Convert interest raet to its decimal equivalent
'  i.e. 12.75 becomes 0.1275
fIntrate = Val (txtIntRate.Text) / 100
'convert annual interest rate to monthly
'  by dividing by 12 (month in a year)
fIntrate = fIntrate / 12
'convert number of years to number of months
' by multiplying by 12 (months a year)
nTerm = Val (txtTerm.Text) * 12
'Calculate and display the monthly payment.
' The Format function makes the displayed number look good.
cPayment = cPrincipal * (fIntrate / (1 - (1 + fIntrate) ^ -nTerm))
txtpayment.Text = Format(cPayment, "Fixed")
```

הרצת התוכנית

בטוח שהינך משתוקק כבר לראות את התוכנית שלך בפעולה! למרות זאת, לפני הרצתה, ודא כי שמרת את עבודתך. כדי להריץ את התוכנית, עליך להפעיל את פקודת ההרצה של Visual Basic באחת משיטות אלו:

- לחץ על לחצן **Start** בסרגל הכלים.
- בחר **Run**, **Start** מתוך התפריט.
- הקש F5.



תרשים 2.19: הכנסת הקוד לחישוב התשלום החודשי משלימה את שגרת האירוע Click עבור cmdCalculate

בביצוע פקודת ההפעלה (Start), Visual Basic מבצעת קומפילציה על התוכנית שלך כדי לחפש שגיאות. אם לא נמצאו כאלו, התוכנית מתחילה להתבצע, ותקבל את הטופס

הראשי. שים לב שההודעה בשורת הכותרת של Visual Basic השתנתה ממצב עיצוב למצב הרצה (Run Mode), דבר המראה שהתוכנית אכן רצה. מכיון שהאפליקציה שבנית היא מונחית עצמים ומונעת על ידי אירועים, היא מחכה שאתה (המשתמש) תיצור אירוע כגון הקלדה בתיבת טקסט או לחיצה על לחצן.

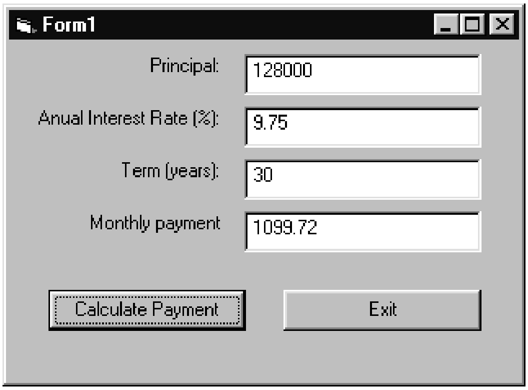
בדוק את התוכנית בעזרת הקלדת ערכים לסך כל ההלוואה (Principal), תנאי (Term), ואחוז הריבית (Interest Rate). השתמש בערכים אלה לבדיקה הראשונה:

Principal	128000
Interest Rate	9.75
Term	30

לאחר הכנסת הערכים, לחץ על לחצן Calculate Payment. התשלום החודשי המופיע אמור להיות 1099.72 (ראה תרשים 2.20)

הערה:

מכיון שהתוכנית שבנינו היא דוגמה לטכניקות תכנות בסיסיות, ממשק המשתמש שלה הוא פשוט ביותר. מאמץ מועט מאוד נעשה כדי להפוך את התוכנית לחסינה בפני שגיאות. יש להכניס מספרים ללא פסיקים או סימן דולר כדי שהחישובים יעבדו כראוי. לאחר שנפתח את יכולת התכנות לאורך הספר, תלמד כיצד להתגבר על מגבלות מסוג זה.



תרשים 2.20: תוכנית מחשב ההלוואות פועלת כעת לחלוטין ויכולה לחשב תשלומי הלוואות.

בדוק את התוכנית עם צירופים שונים של מספרים. אחר, סיים את התוכנית בעזרת לחיצה על לחצן הסיום (End) בסרגל הכלים של Visual Basic. פעולה זו תחזיר אותך לסביבת הפיתוח.

בתקליטור:

תוכנית מחשב ההלוואות המלאה נמצאת בתקליטור, בתיקיה
Books\59221/Chap02\LoanCalc.

מכאן...

תוכנית מחשב ההלוואות איפשרה לנו לצלול לתוך כתיבת תוכנית Visual Basic מאפס. בפרק זה למדת לבצע את הפעולות הבאות:

- להפעיל את Visual Basic.
- להתחיל פרויקט חדש.
- להוסיף פקדים לטופס.
- לשנות את המאפיינים של האובייקטים בפרויקט.
- כתיבת קוד בכדי להפיח בתוכנית חיים.

האמן או לא, אך פעולות אלו מהוות את גרעין התכנות ב- Visual Basic. אומנם ראית דוגמאות פשוטות של כל אחד מן הצעדים האלה, אך רוב התכנות מהווה חזרה על צעדים אלה פעמים רבות עד לקבלת התוצאה הרצויה.

כשנתחיל לעבוד על יישומים מסובכים יותר, תראה דרכים רבות לתרגול ולחיזוק יסודות אלה של תכנות ב- Visual Basic. פנה לפרקים אלה כדי לקבל מידע רלוונטי נוסף:

- כדי לראות כיצד להשתמש במרכיבי היסוד של Visual Basic, כדי ליצור יישומים משלך ראה פרק 3 **אבני היסוד של Visual Basic**.
- כדי ללמוד אודות הפקדים השונים הזמינים לנו לשם בניית תכניות, ראה פרק 4 **שימוש בפקדי ברירת המחדל של Visual Basic**.
- כדי ללמוד עוד על כתיבת קוד ב- Visual Basic, כדי לתקשר עם משתמשי התוכנית שלך, ראה פרק 5 **תגובה באמצעות שגרות אירוע**.

3

אבני היסוד של Visual Basic

בפרק זה?

- טפסים
- השימוש בפקדים
- חקר המאפיינים
- מבט ראשון על שיטות ואירועים
- מבט חוזר על מאפייני הטופס

כבר למדנו שאובייקטים, למשל טפסים ופקדים, הם מרכיבים של Visual Basic אשר אתם מתקשר המשתמש. באופן כללי האובייקטים שהמשתמש רואה ומשתמש בהם נקראים **המרכיבים הוויזואליים** (Visual Components) של התוכנית, לעומת **מרכיבי הקוד** (Code Components) של התוכנית שמתייחסים לקוד התוכנית שהמפתח יצר.

בפרק זה נבחן כמה מהיסודות הנחוצים לבניית המרכיבים הוויזואליים של התוכנית. נלמד כיצד להגדיר את צורת ההופעה והתנהגות אובייקט על ידי שינוי **המאפיינים** (Properties), **השיטות** (Methods) **והאירועים** (Events) שלו. הידיעה כיצד לנהל את מרכיבי האובייקטים הללו, תציב אותך על המסלול הנכון בדרך ליצירת ממשקי משתמש מקצועיים בעצמך.

הערה:

המונח **אובייקט** (Object) בפרק זה מתייחס לאובייקטים ויזואליים כגון טפסים ופקדים. בפרקים הבאים נדון בפירוט רב יותר באובייקטים אלה, וכן באובייקטים אחרים ובדרך שבה הם מתייחסים לקוד של Visual Basic.

טפסים

רוב הדוגמאות, בהן נתקלת עד כה, עשו שימוש באובייקט הטופס. **הטופס**, ובאנגלית - Form, מהווה מעין "מחסן" עבור כל שאר הפקדים, כגון - פקד Label (תווית), פקד TextBox (תיבת טקסט) ופקד Image (דמות), היוצרים את ממשק המשתמש שלך. מרבית התוכניות שתבנה תכלולנה מספר טפסים.

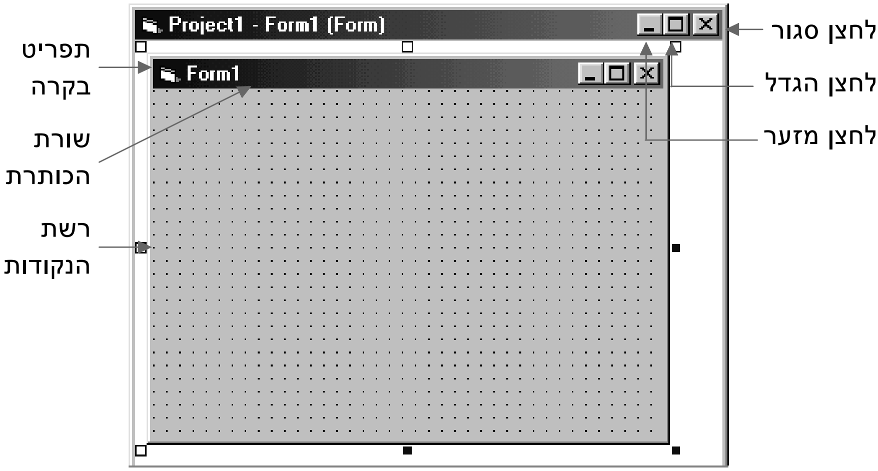
הערה:

באפשרותך אף ליצור תוכנית Visual Basic אשר אינה מכילה טפסים כלל! דוגמה טובה לכך היא תוכנית שורת-פקודה המעבדת קבצים ואינה זקוקה לממשק משתמש.

חלקי הטופס

כאשר הינך מתחיל לעבוד על פרויקט חדש מסוג Standard EXE, Visual Basic טוענת עבורך פרויקט ברירת מחדל המכיל, במצב רגיל, טופס סטנדרטי יחיד המכונה Form1 (ראה תרשים 3.1). מאחר וזה המוצא של תהליך יצירת ממשק המשתמש לתוכניתך, מומלץ כי תשקיע מעט זמן בלימוד חלקיו השונים של הטופס.

כפי שאתה יכול לראות בתרשים 3.1, טופס Visual Basic מכיל את כל האלמנטים שאתה עשוי לצפות להם בתוכנית Windows. הוא מכיל שורת כותרת, תפריט בקרה, ואוסף לחצני הגדל/שחזר, מזער ולחצן סגירה. שים לב כי בזמן בניית התוכנית מרבית מרכיבי הטופס זמינים, זאת למרות שעל פי הגדרת המאפיינים של חלקם (לדוגמה לחצן הסגירה), הם אינם זמינים בזמן פעולת התוכנית.



תרשים 3.1: טופס ריק הוא נקודת המוצא לבניית ממשק המשתמש

תכונה נוספת המאפיינת את זמן התכנון היא רשת הנקודות המאפשרת הצבת פקדים באופן מדויק בזמן תכנון ממשק המשתמש. אתה יכול לשלוט בהתנהגות רשת התכנון דרך תיבת הדו-שיח **Options**, אליה ניתן להיכנס על ידי בחירת **Tools** ואז **Options**, מתוך התפריט הראשי. תיבה זו מאפשרת לשנות את גודל הרשת או אף לבטל אותה כליל. כמו כן, ניתן לבחור ביישור אוטומטי של הפקדים על פי קווי הרשת. כאשר אפשרות זו מופעלת (כברירת מחדל), הפינה השמאלית-עליונה של כל פקד מיושרת ביחס לנקודת הרשת הקרובה ביותר. אופציית ברירת מחדל זו מאפשרת סידור קל ונוח של הפקדים על גבי הטופס. אנו אף מעדיפים לצופף את קווי הרשת על גבי הטופס, דבר המאפשר יישור מדויק יותר של הפקדים.

מה עושים טפסים?

כפי שהזכרתי קודם לכן, טפסים מייצגים את ממשק המשתמש בתוכנית שלך. טפסים הם למעשה חלק התוכנית עמו מתקשר המשתמש. כל הפקדים שעימם עובדים המשתמשים, כמו פקד TextBox (תיבת טקסט), פקד Command Button (לחצן פקודה) וכיוצא בזה, ממוקמים על גבי אחד או יותר מהטפסים.

בנוסף לתפקידם כמחסן לפקדים, מהווים הטפסים מעין "שוטרי תנועה" המפקחים על האובייקטים השונים המרכיבים את הממשק. בהמשך תוכל לראות דוגמאות רבות לשימוש בקוד המגיב לאירועים המתייחסים לאובייקטים השונים. כל אחד מקטעי הקוד בתוכנית, המתייחס לאובייקטים הממוקמים על גבי טופס מסוים, מאוחסן (ונערך) כחלק מהטופס עצמו. במילים אחרות, הטופס מכיל לא רק את האובייקטים שהמשתמש **יכול** לראות, אלא גם את קטעי הקוד שהמשתמש **אינו יכול** לראות.

השימוש בפקדים

למרות שטפסים הם חלק חשוב בתוכנית שלך, אין להם ערך רב ללא הפקדים המצורפים אליהם. פקדי Visual Basic מאפשרים לך לבצע מיגוון רחב של משימות, כגון עריכת טקסט, הצגת תמונות ותקשורת עם בסיסי נתונים. השימוש הליברלי בפקדים היה ונשאר הצד החזק של Visual Basic.

מהם פקדים?

ה**פקדים** (Controls) של Visual Basic הם אובייקטים המיועדים לביצוע משימה כלשהי. לפקדים יש מאפיינים, אירועים ושיטות המשויכים אליהם, בדומה לטופס עצמו. לדוגמה, באמצעות מאפייני פקד TextBox (תיבת טקסט), נקבע את גודל תיבת הטקסט, את סוג הגופן בו יופיע הטקסט בתיבה, את צבע הטקסט, הרקע שלו ועוד.

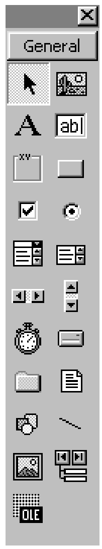
Visual Basic הינה שפה גמישה המאפשרת למפתח לשלב בתוכנית גם פקדים שאינם בהכרח מבית היוצר של חברת Microsoft. לאור גמישותה של Visual Basic והמיגוון העצום של פקדי "צד שלישי" בשוק, קיים סיכוי רב שתוכל לאתר פקד עבור כל משימה בה תחפוץ, החל מאיחזור מידע והפקת דוחות מותאמים אישית, וכלה בפקדים המתמחים בעיבוד גרפי עבור משחקים, וכל דבר אחר שעולה על הדעת.

השימוש בפקדים לביצוע משימות קיים מאז תחילת ימי Visual Basic, אך היכולת ליצור ולשלב משלך שולבה לראשונה בגירסה 5. כעת Visual Basic מאפשרת לבנות פקדי ActiveX עבור תוכניותיך או עבור כל תוכנית אחרת התומכת בתקן ActiveX.

פונקציות הפקדים

אפשר לחשוב על פקד כעל תוכנית זעירה המבצעת משימות ייחודיות. הפקד TextBox, לדוגמה, מציג טקסט בגודל ובגופן על פי הערכים שהכנסנו למאפייניו. בנוסף, מכיל, אותו פקד, קוד פנימי המאפשר לו לקלוט ולעבד לחיצות מקשים באופן שבו הוא יודע, לדוגמה, שלחיצה על מקש Backspace משמעותה כי עליו למחוק את האות האחרונה שהוקשה. אם התנסית בכתיבת תוכניות בשפות תכנות אחרות, ובעיקר בשפות הפעולות בסביבת הפיתוח של DOS, אתה בוודאי יודע שלעיתים קרובות יש לכתוב קטע קוד גדול למדי רק כדי לקלוט ולעבד לחיצות מקשים של המשתמש, ובכך לאפשר לו להזין נתונים לתוכנית. ב-Visual Basic, לעומת זאת, כל שעליך לעשות הוא למקם את הפקד על גבי הטופס וכל היתר כבר ייעשה עבורך. במילים אחרות, פקד הוא אובייקט מוגדר מראש המבצע עבורך פעולות מיוחדות.

גירסה 6 של Visual Basic מצוידת במערכת פקדים סטנדרטיים, אותה ניתן למצוא גם בשאר המהדורות, אשר מאפשרת לך לטפל בסוגים רבים של משימות תכנות. פקדים אלה מופיעים ב**ארגז הכלים** (ToolBox) המוצג בתרשים 3.2. מטרת כל פקד בארגז הכלים ואופן השימוש בו מתוארים בפרק 4 **שימוש בפקדי ברירת המחדל של Visual Basic**.



תרשים 3.2: ארגז הכלים של Visual Basic מכיל אוסף פקדים סטנדרטי

הערה:

הפקדים הסטנדרטים המופיעים בארגז הכלים, כפי שמוצג בתרשים 3.2, אינם הפקדים היחידים המצורפים ל- Visual Basic. כדי להוסיף פקדים נוספים כגון פקד Treeview או פקד ImageList, יש להשתמש בתיבת הדו-שיח של הרכיבים המופיעה ב- **Components** תחת **Project** מהתפריט הראשי.

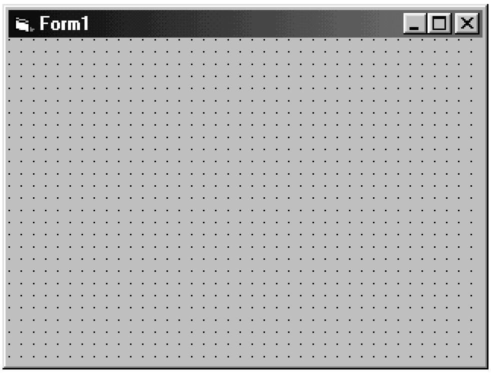
ראה: פקד TreeView (מבנה עץ) בפרק 12 **הפקדים המשותפים של Microsoft.**

חקר המאפיינים

בפרק 2 **יצירת התוכנית הראשונה**, למדת כיצד לשנות את מאפייני הפקדים כדי לשנות את צורת הופעתם בעת הפעלת התוכנית. חשוב על המאפיינים כעל תואר שם (Adjective) המתאר תכונה ספציפית של אותו אובייקט. ניתן להתאים בקלות את צורת הופעת האובייקט לזו בה אנו מעוניינים, על ידי שינוי ערך מאפיין אחד או יותר.

יסודות המאפיינים

כאשר הינך מסתכל על טופס כלשהו, אתה רואה למעשה חלון מלבני, כמו זה המופיע בתרשים 3.3. צורת הופעתו של חלון הטופס נקבע על ידי אוסף מאפיינים. לדוגמה, מיקום הטופס על גבי המסך נקבע באמצעות המאפיינים Left ו-Top, גודל הטופס באמצעות המאפיינים Width ו-Height שלו, והכותרת באמצעות המאפיין Caption (כותרת). ניתן אף לקבוע מהם לחצני הבקרה שיופיעו (אם בכלל) בשורת הכותרת של הטופס, זאת על ידי שינוי ההגדרות של מספר מאפיינים במקביל.



תרשים 3.3: מראה הטופס נקבע על ידי מאפייניו

בעת שמירת הטופס כחלק מפרויקט, Visual Basic יוצרת קובץ טקסט בעל סיומת FRM. קובץ זה מכיל בתוכו אינפורמציה על הטופס, הכוללת את מאפייני הטופס, האובייקטים הכלולים בו ומאפייניהם, וכן את כל קטעי קוד שנוצר עבור טופס זה. תרשים 3.4 מציג חלק מקובץ FRM. שנוצר עבור הטופס המופיע בתרשים 3.3.

```
VERSION 5.00
Begin VB.Form Form1
   Caption         =   "Form1"
   ClientHeight    =   3840
   ClientLeft      =   60
   ClientTop       =   345
   ClientWidth     =   3885
   LinkTopic       =   "Form1"
   ScaleHeight     =   3840
   ScaleWidth      =   3885
   StartUpPosition =   3  'Windows Default
   Begin VB.CommandButton cmdWiden
      Caption         =   "Widen Form"
      Height          =   495
      Left            =   1320
      TabIndex        =   8
      Top             =   2880
      Width           =   1695
   End
   Begin VB.Label lblTop
      BeginProperty Font
         Name            =   "MS Sans Serif"
```

תרשים 3.4: קובץ FRM. של טופס מכיל מידע על המאפיינים שמגדירים אותו ואת האובייקטים הכלולים בו



אזהרה:

עבור טפסים מסוימים, Visual Basic יוצרת בנוסף גם קובץ בעל סיומת FRX. בקובץ נשמרים אלמנטים גרפיים ובינאריים שלא ניתן לאחסן בקובץ טקסט רגיל. אם ברצונך להעתיק טופס מתיקיה אחת לאחרת, חשוב מאוד להעתיק גם את קובץ FRX. הקשור אליו.

מאפיינים נפוצים

לא לכל האובייקטים ב- Visual Basic יש אוסף מאפיינים זהה. עם זאת, יש מספר מאפיינים שכיחים המופיעים באובייקטים רבים. ברשימה שלהלן מופיעים חלק מן המאפיינים השכיחים והחשובים ביותר:

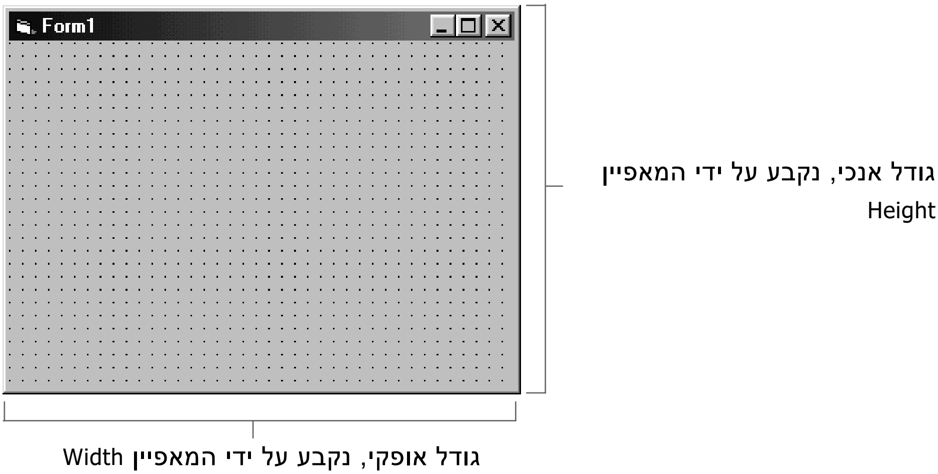
- Name
- Index
- Left
- Top
- Height
- Width
- Enabled
- Visible

שימוש במאפיינים לשליטה בגודל אובייקט

גודל אובייקט ב- Visual Basic נקבע על ידי מאפייני Height ו-Width שלו. כפי שכבר ראית, ניתן לשנות את גודל האובייקט על ידי סימון וגרירת נקודות האחיזה שלו בזמן עיצוב, או על ידי שינוי ערכי המאפיינים Height ו-Width בזמן התכנון או בזמן פעולת התוכנית. שינוי גודל האובייקט בזמן עיצוב, גורם גם לשינוי ערכי Height ו-Width בחלון המאפיינים, כפי שגילית זה מכבר בפרק 2.

ניתן להשתמש בקוד כדי לגרום לשינוי בגודל האובייקט במהלך פעולת התוכנית. במקרים מסוימים, לדוגמה כשמדובר בטופס, יכול המשתמש עצמו לשנות את גודל האובייקט, זאת על ידי גרירת אחת הפינות שלו. בסעיף **שינוי מאפיינים בזמן פעולת התוכנית** תוכל לראות דוגמה לעקרון זה.

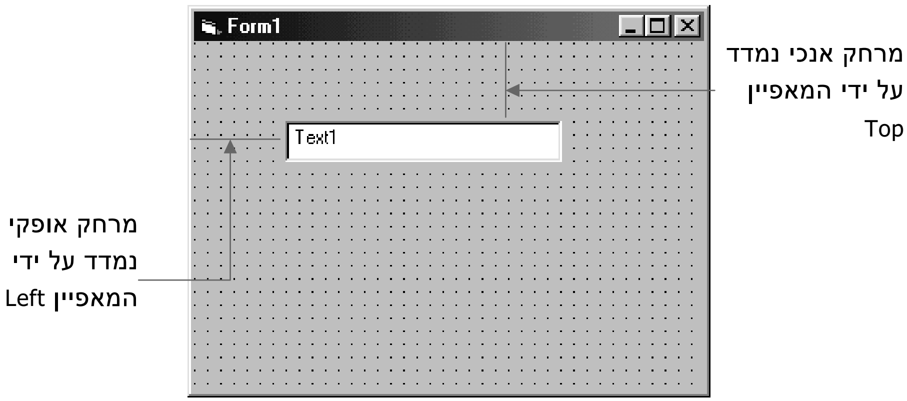
תרשים 3.5 ממחיש כיצד באים לידי ביטוי מאפייני הגודל Height ו-Width של הטופס.



תרשים 3.5: גודלו של טופס נקבע בהתאם למאפיינים Height ו-Width

שימוש במאפיינים לשליטה במיקום אובייקט

בנוסף לשליטה בגודל האובייקט, אתה יכול לשלוט גם במיקומו, זאת בעזרת המאפיינים Top ו-Left (ראה תרשים 3.6). המאפיין Left מייצג את המרחק שבין צידו השמאלי של האובייקט לצידו השמאלי של האובייקט המכיל אותו. המאפיין Top מייצג את המרחק שבין קצהו העליון של האובייקט לקצהו העליון של האובייקט המכיל אותו. כשמדובר בטופס עצמו, ה"אובייקט המכיל" הוא המסך כולו. אם תצייר פקד על גבי הטופס, הטופס הוא האובייקט המכיל של הפקד. ראוי לציין כי יש מספר פקדים, כמו לדוגמה PictureBox (תמונה) ו-Frame (מסגרת), המסוגלים להכיל פקדים אחרים.

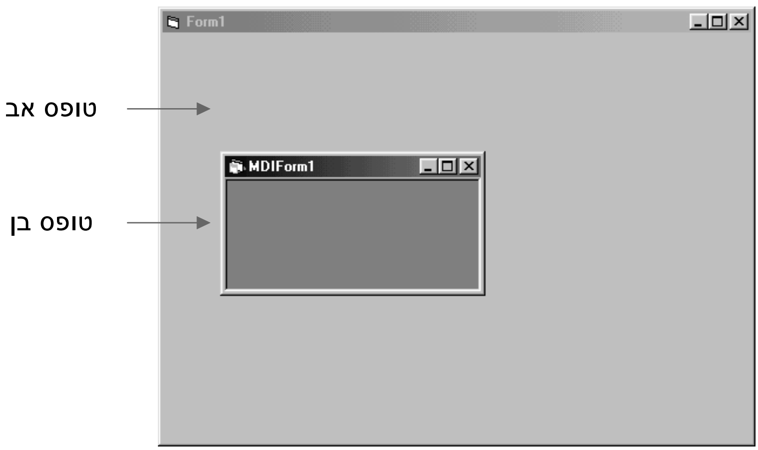


תרשים 3.6: מיקום הפקד TextBox נמדד ביחס לטופס המכיל אותו

הערה:

המאפיינים Top ו-Left מסוגלים להכיל גם ערכים שליליים. לדוגמה, שיבוץ הערך 1440- במאפיין Left של פקד Label, יגרום למיקום הפקד כך שהקצה השמאלי שלו ייִמצא כ-2.5 ס"מ משמאל לאובייקט המכיל אותו. לפיכך לא נוכל לראותו (במלואו או בחלקו).

שלא כרוב הטפסים שמיקומם נמדד ביחס לפינת השמאלית-עליונה של המסך, מיקום טופס שהוא חלק מממשק **מרובה מסמכים** (Multiple Document Interface - MDI) נמדד ביחס לפינה השמאלית-עליונה של אזור הלקוח בטופס האב (ראה תרשים 3.7).



תרשים 3.7: טופס הצאצא בממשק MDI ממוקם ביחס לטופס האב שלו

שינוי המאפיינים במהלך פעולת התוכנית

כפי שכבר צוין, ניתן לשנות את מאפייני האובייקטים בזמן ריצת התוכנית, אם על ידי פעולות המשתמש או על ידי פקודות המשולבות בקוד. בשלב זה נוכל לבחון את העניין באמצעות תרגיל פשוט: נבנה פרויקט פשוט אשר יציג בפני המשתמש טופס רגיל. המשתמש יוכל לשנות את גודלו ומיקומו של הטופס על ידי גרירת אחד מקצותיו באמצעות העכבר. כמו כן נמקם על גבי הטופס שני לחצני פקודה ונכתוב פקודות קוד אשר תגרומנה להרחבת הטופס בעת לחיצה על הלחצן הראשון, ושינוי מיקומו כלפי מטה בעת לחיצה על הלחצן השני. בנוסף לכך, נמקם על גבי הטופס גם ארבעה פקדי Label אשר יציגו את ערכי המאפיינים Width (רוחב), Height (גובה), Top (מרחק למעלה) ו-Left (מרחק שמאלה).

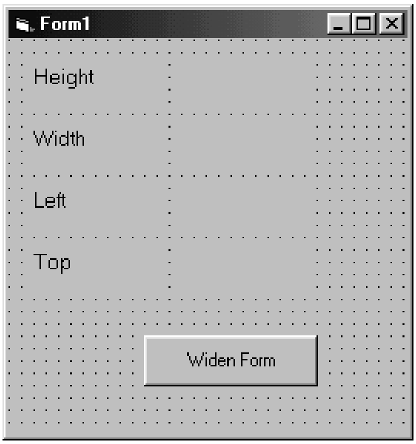
יצירת תוכנית המשנה את גודל ומיקום הטופס

התייחסות למאפייני האובייקטים, במסגרת קוד התוכנית, נעשית בשיטה המכונה **זיהוי על פי נקודות** (Dot Notation). לדוגמה, המאפיין Caption של פקד Label כלשהו הממוקם בטופס מסוים ונקרא lblAddress, ייכתב בקוד בצורה הבאה: lblAddress.Caption.

אם אינך זוכר כיצד למקם אובייקטים על גבי הטופס או כיצד לשנות את מאפייניהם, כדאי שתערוך חזרה קצרה על פרק 2.

בתרשים 3.8 מופיע טופס הדוגמה כפי שהוא אמור להיראות בסוף התהליך. ייתכן ותרצה להיעזר בתרשים זה במהלך בניית הפרויקט שלך.

ראה: יצירת ממשק המשתמש לתוכנית בפרק 2 **יצירת התוכנית הראשונה**.



תרשים 3.8: פרויקט דוגמה זה מדגים את השימוש בקוד לשינוי מאפייני טופס שונים

בתקליטור:

הפרויקט מצורף בתקליטור ונקרא FormProp.

כדי לבנות גירסה משלך לפרויקט זה, בצע את הפעולות הבאות:

1. צור פרויקט חדש מסוג Standard EXE.
2. הוסף ארבעה פקדי Label לטופס. סדר אותם בטור בצדו השמאלי של הטופס. פקדים אלה ישמשו להצגת מידע במסגרת התוכנית. בפקדי Label המשמשים אך ורק להצגת טקסט, אין צורך, בדרך כלל, לשנות את ערך המאפיין Name, שכן ממילא לא תהיה התייחסות אליו במסגרת הקוד. שנה את מאפיין Caption של ארבעת הפקדים ל: Height, Width, Left, Top (בהתאם).
3. הוסף ארבעה פקדי Label נוספים, מימין לטור הקודם. שנה את מאפיין Name שלהם ל-lblHeight, lblWidth, lblLeft, lblTop (בהתאם). השאר את מאפיין Caption שלהם ריק (עליך למחוק את הערך המשובץ כברירת מחדל ב-Caption).
4. הוסף פקד Command Button (לחצן פקודה) בחלק התחתון של הטופס. שנה את המאפיין Name ל-cmdWiden ואת מאפיין Caption ל- Widen Form.
5. סמן את הטופס ושנה את המאפיין Width ל-4000 Twips (ראה סעיף **מידות ב-Visual Basic** להגדרת המושג Twips).
6. עבור לחלון הקוד על ידי לחיצה כפולה על לחצן הפקודה. כעת הסמן אמור להיות ממוקם בשגרת המשנה cmdWiden_Click.
7. הקש טאב והקלד: Form1.Width = Form1.Width + 100. שורת קוד זו גורמת לערך של המאפיין Width של Form1 לגדול ב-100 Twips בכל פעם שלחצן הפקודה נלחץ.
8. פתח את תיבת האובייקט של חלון הקוד ובחר Form. פתח את תיבת האירועים של חלון הקוד ובחר Resize. פעולה זו יוצרת את המעטפת של שגרת האירוע Form_Resize. האירוע Resize מתרחש בכל פעם שרוחב או אורך הטופס משנים את גודלם, אם על ידי המשתמש ואם על ידי קוד התוכנית.
9. הוסף את שורות הקוד הבאות לשגרת האירוע Form_Resize:

```
lblWidth.Caption = Form1.Width
lblHeight.Caption = Form1.Height
lblLeft.Caption = Form1.Left
lblTop.Caption = Form1.Top
```

10. אם אתה מעוניין, תוכל לשמור את הטופס והפרויקט, אולם אין הדבר הכרחי.



טיפ:

Visual Basic מצוידת בכלי שימושי לעזרה בכתיבת מאפיינים הנקרא Auto List Members. כאשר הינך מקליד שם אובייקט מסוים ואחריו נקודה, Visual Basic מנסה לעזור לך בהשלמת הקוד החסר על ידי הצגת רשימת מאפיינים המתאימים לסוג האובייקט אותו הקלדת (בנוסף לאלמנטים נוספים, כגון שיטות ואירועים, המתאימים גם הם לאובייקט זה). שים לב כי הקלדת התווים הראשונים מתוך שמו של המאפיין המבוקש גורמת להדגשתו ברשימת המאפיינים. לחיצה בשלב זה על רווח, נקודה, הקשת Enter או Ctrl+Enter תגרום להוספת המאפיין הרצוי לשורת הקוד.

מידות ב - Visual Basic

מרחקים ב- Visual Basic נמדדים לפי ברירת המחדל ב**טוויפים** (Twips). זו יחידת מידה עצמאית השווה ל- 1/20 מנקודת מדפסת. לכן יש 1440 טוויפים באינץ'. הגודל הפיסי האמיתי שלה משתנה בהתאם לרזולוציית המסך. ניתן להחליף את יחידות המידה המשמשות למיקום ומדידת אובייקטים ביחס לאובייקט המכיל אותם על ידי שינוי הערך במאפיין ScaleMode של האובייקט המכיל. אך לא ניתן לשנות את יחידות המידה של המסך. לכן המאפיינים Height, Width, Top, Left של טופס תמיד יימדדו בטוויפים.

בדיקת התוכנית שזה עתה יצרת

הפעל את התוכנית שזה עתה יצרת. שנה את גודל הטופס על ידי גרירת קצותיו. שים לב כיצד פקדי Label מציגים את השינויים החלים באורכו ורוחבו של הטופס. פעולה זו מראה כיצד מאפייני אובייקט יכולים להשתנות בעקבות פעולות המשתמש בזמן פעולת התוכנית.

עתה, לחץ על לחצן הפקודה שעל הטופס כדי לגרום להרחבת הטופס. שים לב לשינוי המיידי בתווית הרוחב ולהופעת ערכו החדש של המאפיין Width. דוגמה זו ממחישה שינוי מאפיינים באמצעות פקודות תכנות (למרות שהמשתמש גרם לשינוי הרוחב, השינוי אירע למעשה בעקבות ביצוע שורת קוד).

לבסוף, נסה להזיז את הטופס (על ידי גרירת שורת הכותרת שלו). על ידי כך אתה (כמשתמש) משנה את ערכי המשתנים Top ו-Left. דוגמה זו בעייתית מעט מכיון שהתוויות הרלבנטיות אינן מתעדכנות בערכים החדשים של המאפיינים Top ו-Left. הדבר קורה מכיון שהקוד אשר מטפל בעדכון פקדי Label נמצא בשגרת האירוע Resize. מכיון שלא שינינו את גודל הטופס אלא רק הזזנו אותו, העדכון לא התבצע. למרבה הצער אין אירוע, בדומה ל-Resize, המתרחש בעת הזזת הטופס. בעיה זו ניתנת כמובן לפתרון, אך הפתרון נמצא מעבר להיקף החומר המכוסה על ידי פרק זה. בינתיים אפשר לעדכן את הערך "ידנית" על ידי שינוי גודל הטופס מייד לאחר הזזתו, פעולה המאפשרת לך לראות את הערכים החדשים של Top ו-Left.

שימוש במאפיינים לשליטה על האינטראקציה עם המשתמש

גם אם בנית אפליקציה המכילה טפסים ופקדים רבים, סביר להניח כי לא תרצה לאפשר למשתמש גישה בו-זמנית לכולם. אם, לדוגמה, ברצונך לבנות יישום המאפשר למשתמש להקליד דוח הוצאות, תרצה שיישום זה יכיל לחצן המאפשר למשתמש לשלוח את הדוח המוכן להדפסה, אך לא תרצה לאפשר למשתמש לגשת ללחצן זה בטרם יהיו כל הדוחות מוכנים ומאוזנים. שני המאפיינים, Visible (גלוי) ו-Enabled (מופעל), יעזרו לך לשלוט על האינטרקציה עם המשתמש.

באמצעות המאפיין Visible אנו קובעים האם אובייקט מסוים יופיע על גבי המסך או לא. באמצעות המאפיין Enabled אנו קובעים האם תתאפשר למשתמש הגישה לאובייקט מסוים או לא. שני המאפיינים יכולים להכיל אחד משני ערכים: True או False.

כאשר המאפיין Visible נמצא במצב False, האובייקט לא יופיע על המסך והמשתמש אף לא יידע על קיומו. כאשר המאפיין Enabled נמצא ב-False, האובייקט יופיע על המסך (במידה המאפיין Visible נמצא ב-True), אך במצב מנוטרל, כך שלא ניתן יהיה להשתמש בו. אובייקט מנוטרל מופיע על המסך בגוון אפור או במעומעם, כדי שהמשתמש יידע שהוא שם אך אינו ניתן לשימוש.

דוגמה טובה לאובייקטים זמינים ובלתי זמינים חליפות, ניתן למצוא בממשק האשפים (Wizards) במספר תוכנות Windows. האשפים מפרידים משימה מסוימת למספר שלבים הגיוניים. המעבר בין שלבי האשף מתבצע בעזרת שלושה לחצני ניווט (בדרך כלל Back, Next ו-Finish). זמינות לחצני הניווט משתנה בהתאם לשלב אליו הגיע המשתמש, כפי שמתואר בתרשים 3.9.



תרשים 3.9: כיון שהמשתמש באשף זה נמצא בשלב 1, מאפיין Enabled של הלחצן Back נמצא במצב False, ולכן הלחצן מופיע באפור

טיפ:

שיטה אחת בה אתה יכול להיעזר, אם ברצונך לבנות ממשק אשף ב- Visual Basic, היא לצייר את הפקדים עבור כל שלב בתוך מסגרת. הפקד Frame (מסגרת) של Visual Basic משמש כ"מחסן" לפקדים אחרים. לפיכך, כל שינוי במאפיין Visible שלו משפיע גם על מצב הופעתם של הפקדים המאוחסנים בו.

אופן ההפניה לטפסים ופקדים בקוד

מרכיב מרכזי נוסף בכל אובייקט Visual Basic הוא המאפיין Name. מאפיין Name מגדיר ערך מזהה ייחודי שעל פיו תוכל לפנות לאובייקט זה בקוד. לכל טופס, או פקד חייב להיות שם ייחודי.

הערה:

לכל טופס בפרויקט חייב להיות שם ייחודי. לעומת זאת שמות פקדים חייבים להיות ייחודיים רק עבור הטופס שבו הם נמצאים. לדוגמה, יכול להיות לך פקד בשם Text1 בכל אחד מהטפסים שבפרויקט שלך, אך לא יכולים להיות לך שני טפסים שייקראו Form1.

ברגע שאובייקט נוצר, Visual Basic נותנת לו שם ברירת מחדל. לדוגמה, Form1 הוא השם שניתן לטופס הראשון שתיצור בפרויקט שלך. למרות זאת, הדבר הראשון שעליך לעשות לאחר יצירת אובייקט חדש הוא שינוי שמו לשם בעל משמעות. אנו, לדוגמה, משתמשים לעיתים קרובות בשם frmMain עבור טופס הממשק הראשי בפרויקט.

כפי שמוזכר בפרק 2, שמירה על סטנדרט אחיד במתן שמות לטפסים ואובייקטים אחרים הוא הרגל טוב למפתח (סטנדרט זה נקרא גם קוד הונגרי). נהוג להשתמש בקיצור בן שלוש אותיות קטנות של שם הפקד כדי שניתן יהיה לזהות את סוג האובייקט בקלות. יתרת השם מתארת את מטרת האובייקט. בשם frmMain, שהוזכר קודם לכן, מרמזת הקידומת frm כי האובייקט הוא טופס, בעוד הסיומת Main מעידה על כך שמדובר בטופס הראשי בתוכנית. בטבלה 3.1 תוכל למצוא המלצות עבור שמות אובייקטים רבים ב- Visual Basic (טפסים ופקדים).

זכור כי השמות שאתה מעניק לאובייקטים משמשים גם במסגרת הקוד, כך שרצוי, למען הנוחות, להעניק שמות בעלי אורך סביר.

כדי לשנות את המאפיין Name של אובייקט כלשהו, היכנס לחלון המאפיינים (באחת משלוש הדרכים הבאות: לחץ על לחצן Properties, בחר View, Properties Window או הקש F4 מתוך החלון הראשי), ובחר את המאפיין Name. הקש ערך חדש למאפיין. תרשים 3.10 מראה את מאפיין Name בחלון המאפיינים.

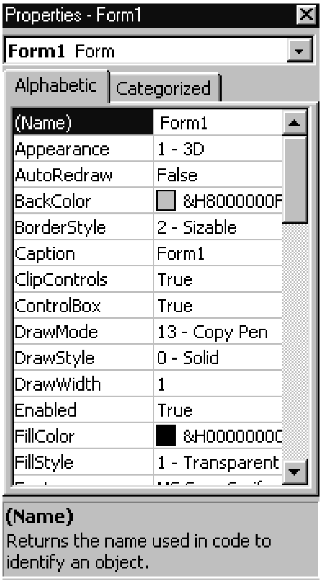
הערה:

ב- Visual Basic, חלון המאפיינים מוצג על המסך כברירת מחדל, כך שאין צורך לפתוח אותו במיוחד, אלא אם כן סגרת אותו.

טבלה 3.1: אובייקטים ב- Visual Basic והקידומות שלהם

סוג האובייקט	קידומת
CheckBox (תיבת סימון)	chk
Combo box (תיבה משולבת)	cbo
Command Button (לחצן פקודה)	cmd
Common Dialog (תיבת דו-שיח משותפת)	cdl
Data control (פקד מידע)	dat
Data bound combo box (תיבה משולבת מקושרת מידע)	dbc
Data Bound Grid (רשת מקושרת מידע)	dbg
Data Bound ListBox (תיבת רשימה מקושרת מידע)	dbl
Directory List Box (תיבת רשימה לספריה)	dir
Drive ListBox (תיבת רשימה לכונן)	drv
File ListBox (תיבת רשימה לקובץ)	fil
Form (טופס)	frm
Frame (מסגרת)	fra
Grid (רשת)	grd

סוג האובייקט	קידומת
Horisontal scrollbar (פס גלילה אופקי)	hsb
Image (תמונה)	img
Label (תווית)	lbl
Line (קו)	lin
List Box (תיבת רשימה)	lst
Menu (תפריט)	mnu
OLE Container (מאחסן OLE)	ole
Option Button (לחצן אפשרות)	opt
Picture Box (תיבת תמונה)	pic
Shape (צורה)	shp
TextBox (תיבת טקסט)	txt
Timer (מודד זמן)	tmr
Vertical ScrollBar (פס גלילה אנכי)	vsb



תרשים 3.10: המאפיין Name ממוקם בקצה העליון של הרשימה האלפביתית וכמו כן מופיע כמאפיין הראשון בכרטיסיה Categorized תחת הקטגוריה Misc.

מבט ראשון בשיטות ואירועים

פרק זה התרכז, עד כה, במאפיינים וכיצד הם משמשים לשליטה בצורת הופעתם של האובייקטים. בנוסף למאפיינים, אובייקט יכול להכיל גם **שיטות** (Methods), המגדירות את המשימות שהאובייקט יכול לבצע. המשימות יכולות להיות פשוטות, כגון הזזת אובייקט למיקום שונה, או מסובכות יותר כגון עדכון מידע בבסיס נתונים.

עבודה עם שיטות

שיטה היא למעשה פונקציית תוכנית הבנויה בתוך האובייקט ומבצעת עבורו פעולה מסוימת. באמצעות השיטה המוטבעת בו, יודע האובייקט כיצד לבצע את המשימה. אין צורך בהוספת הוראה נוספת כלשהי. בטפסים לדוגמה, מוטבעת שיטה, הנקראת PrintFrom, אשר גורמת להדפסת הטופס כפי שהוא נראה על המסך. המשפט Form1.PrintForm יגרום ל- Visual Basic להדפיס עותק של Form1. למעשה, כל הפרטים הקטנים של השיטה PrintForm כבר מוטבעים בתוך אובייקט הטופס, לפיכך, המפתח עצמו אינו צריך לדאוג להם.

כפי שכבר ודאי ניחשת, הפניה לשיטות, כמו למאפיינים, נעשית בשיטת הזיהוי על פי נקודות. Visual Basic משתמשת בתכונת Auto List Members, שצוינה קודם לכן, כדי להציג את רשימת כל השיטות, המאפיינים והאירועים בכל פעם שתקליד שם אובייקט ואחריו נקודה.

אומנם, אובייקטים שונים יכולים להכיל שיטות שונות, אך רבים מהם גם חולקים במשותף את השיטות שלהלן:

- **Drag.** שיטה המטפלת בפעולות בהן המשתמש גורר ומשחרר אובייקטים בתוך שטח האובייקט המכיל אותם.
- **Move.** שיטה המשנה מיקום אובייקט.
- **SetFocus.** שיטה המעבירה את המיקוד אל הפקד הנבחר.
- **Zorder.** שיטה הקובעת האם אובייקט יופיע לפני או מאחורי אובייקטים אחרים בשטח האובייקט המכיל אותם.

הערה:

המערכת מתמקדת בפקד האחרון שהיה בשימוש ומתייחסת אליו כפקד הנוכחי. ניתן להתמקד בפקד אחד בלבד בכל רגע נתון. ניתן לזהות פקד הנמצא במיקוד על ידי סימן העריכה (בפקדי TextBox) או המלבן המקווקו מסביב לפקד (בפקדי - CheckBox, Option Button ו- Command Button)

תגובה לפעולות באמצעות אירועים

בנוסף לביצוע משימות, האובייקטים בתוכניתך יכולים להגיב לפעולות, בין אם נוצרו על ידי המשתמש או נוצרו חיצונית. התגובה לפעולות נעשית על ידי שימוש באירועים. כאשר המשתמש לוחץ, לדוגמה, על לחצן פקודה, הלחיצה יוצרת אירוע מסוג Click. חלק מההגדרות של כל אובייקט הוא אוסף האירועים אשר ניתן לייחס לו ופעולת המשתמש שגורמת להתרחשותם.

דוגמה לפעולת משתמש, הגורמת להתרחשות אירוע, היא לחיצה על לחצן פקודה, בחירת פרט כלשהו מתיבת רשימה או שינוי התוכן בתיבת טקסט. אירועים מתרחשים גם כאשר המשתמש יוצא מטופס או עובר לטופס אחר. כאשר אובייקט יוצר אירוע כלשהו, הוא מבצע **שגרת אירוע** (Event Procedure) עבור האירוע שהתרחש. בכדי שהתוכנית תוכל להגיב לאירועים, עליך להציב קטע קוד בשגרת האירוע הרלוונטית. לדוגמה, בתוכנית ההלוואות, שבנינו בפרק 2, הצבנו קוד בשגרת האירוע Click של לחצן הפקודה.

פרק 5 **תגובה באמצעות שגרות אירוע**, עוסק בפרטים הקטנים של תכנות אירועים. בפרק זה תלמד כיצד לכתוב קוד לטיפול באירועים וכיצד אירועים רבים מתייחסים אחד לשני.

כיצד מאפיינים ושיטות מתקשרים האחד לשני

כעת, אתה כבר יודע שלאובייקטים יש מאפיינים שקובעים את צורת הופעתם, שיטות המאפשרות להם לבצע משימות ואירועים המגיבים לפעולות המשתמש. אתה עלול לחשוב שכל הדברים הללו קורים ללא קשר האחד לשני, אך זה לא תמיד נכון. לפעמים המאפיינים והשיטות של אובייקט מתקשרים אחד לשני. משמעות הדבר היא שכאשר אתה מפעיל שיטה של אובייקט או משנה את תכונותיו בעזרת קוד, אתה עושה זאת כתגובה לאירוע כלשהו.

הערה:

שינויים מסוימים במאפיינים עלולים לגרום להפעלת אירועים. לדוגמה, שינוי מאפייני Height ו-Width בטופס מפעיל את האירוע Resize.

ביטוי לעצמאותם של שיטות ומאפיינים המיוחסים לאובייקטים ניתן לראות בעת שימוש בשיטה Move תוך שינוי ערכי הערכים Top ו- Left. אתה יכול לגרום לאובייקט לשנות מיקום או על ידי שימוש בשיטה Move או על ידי שינוי ערכי Top ו-Left לערכים חדשים. שני קטעי הקוד הבאים גורמים למעשה לביצוע פעולה דומה של הזזת המיקום של TextBox ב-100 טוויפים שמאלה ו-200 טוויפים מלמעלה:

```
'CODE SEGMENT 1 - Move text box by setting its properties
txtName.Left = 100
txtName.Top = 200
'CODE SEGMENT 2 - Move the text box using the move method
txtName.Move 100, 200
```

כדי לראות דוגמה להבדל בין שימוש במאפיינים לבין שימוש בשיטות, הוסף קוד להזזת לחצן הפקודה בפרויקט הקודם. פתח את הפרויקט ובצע את הפעולות הבאות:

1. פתח את חלון הקוד באירוע cmdWiden_Click.
2. הוסף את שורות הקוד הבאות לסוף שגרת האירוע:

```
cmdWiden.Left = 100
cmdWiden.Top = Form1.ScaleHeight - cmdWiden.Height
```

3. הפעל את התוכנית ושנה את גודל הטופס. הקוד ממקם מחדש את לחצן הפקודה באזור תחתית הטופס בעודך משנה את גודלו. הלחצן יהיה תמיד במרחק 100 טוויפים משמאל לטופס.
4. עצור את התוכנית והחלף את שני המשפטים שהוספת בשורת הקוד הבאה:

```
cmdWiden.Move 100, Form1.ScaleHeight - cmdWiden.Height
```

5. הרץ את התוכנית שוב. שים לב שהקוד שונה אך התוצאה זהה. לא משנה אם תשתמש ב-Top ו-Left או ב-Move, הפקד עצמו יטפל בשינוי המיקום עבורך.

אם תקליד את קטעי הקוד הללו, תבחין בוודאי שלשיטה Move יש עוד שני ארגומנטים. ארגומנטים אופציונלים אלה יכולים לשנות את גודל האובייקט. השימוש בשיטה זאת משיג את אותה תוצאה כמו שינוי המאפיינים Top ו-Left לערכים חדשים.

באופן דומה, לשיטות Show ו-Hide של טופס יש תוצאה זהה לשימוש במאפיין Visible. כאשר נפעיל את השיטה Hide, האפקט יהיה זהה לשינוי ערך המאפיין Visible ל-False (דבר שיגרום, כמובן, להעלמת הטופס מהמסך). באופן דומה, השיטה Show משיגה את אותה תוצאה כמו שינוי ערך המאפיין Visible ל-True.

מבט חוזר על מאפייני הטופס

בדומה למרבית האובייקטים של Visual Basic, גם לטפסים יש סדרת מאפיינים השולטים בצורת הופעתם והתנהגותם. בסעיף הקודם, **חקירת המאפיינים**, למדת אודות חלק מהמאפיינים התקפים לגבי הטופס. בסעיף זה, תערוך היכרות עם מספר מאפייני טופס מרכזיים, ובנוסף תלמד כיצד מאפיינים אלה ניתנים לשליטה בזמן תכנון והפעלת התוכנית. טבלה 3.2 מפרטת כמה מהמאפיינים המרכזיים של טפסים, ומספקת הסבר קצר על כל אחד מהם. בנוסף לכך, הטבלה מגדירה האם הערך של המאפיין ניתן לשינוי בזמן ריצת התוכנית.

טבלה 3.2: מאפייני טופס מרכזיים

שם המאפיין	תיאור	ניתן לשינוי בהרצה
BorderStyle	קובע את סוג המסגרת של הטופס	לא
ControlBox	קובע האם תיבת הבקרה (המכילה את תפריטי ההזזה והסגירה) תיראה בזמן הפעלת התוכנית	לא
Font	קובע את סוג הגופן שבו יוצג טקסט בטופס	כן
Icon	קובע מה יהיה הסמל שיוצג בשורת הכותרת של הטופס כאשר הטופס ממוזער	כן
MaxButton	קובע האם הלחצן הגדל (Maximize) יופיע בטופס בזמן פעולתו	לא
MDIChild	קובע האם הטופס הוא צאצא של אפליקציית MDI	לא
MinButton	קובע האם הלחצן מזער (Minimize) יופיע בטופס בזמן פעולתו	לא
StartUpPosition	קובע את מיקומו הראשוני של הטופס כאשר הוא מוצג לראשונה	לא
WindowState	קובע האם הטופס ייראה מוגדל, ממוזער או רגיל	כן

כעת תוכל להביט ביתר פירוט על חלק מהמאפיינים הללו. המאפיין BorderStyle, מכיל שש הגדרות אפשריות השולטות בצורת הופעתה של מסגרת הטופס (ראה טבלה 3.3). באמצעות הגדרות אלו ניתן לשלוט באפשרויות ההגדלה או ההקטנה של הטופס על ידי גרירת גבולותיו, ואף לשלוט בגובה שורת הכותרת (ראה תרשים 3.11).

טבלה 3.3: הגדרות המאפיין BorderStyle

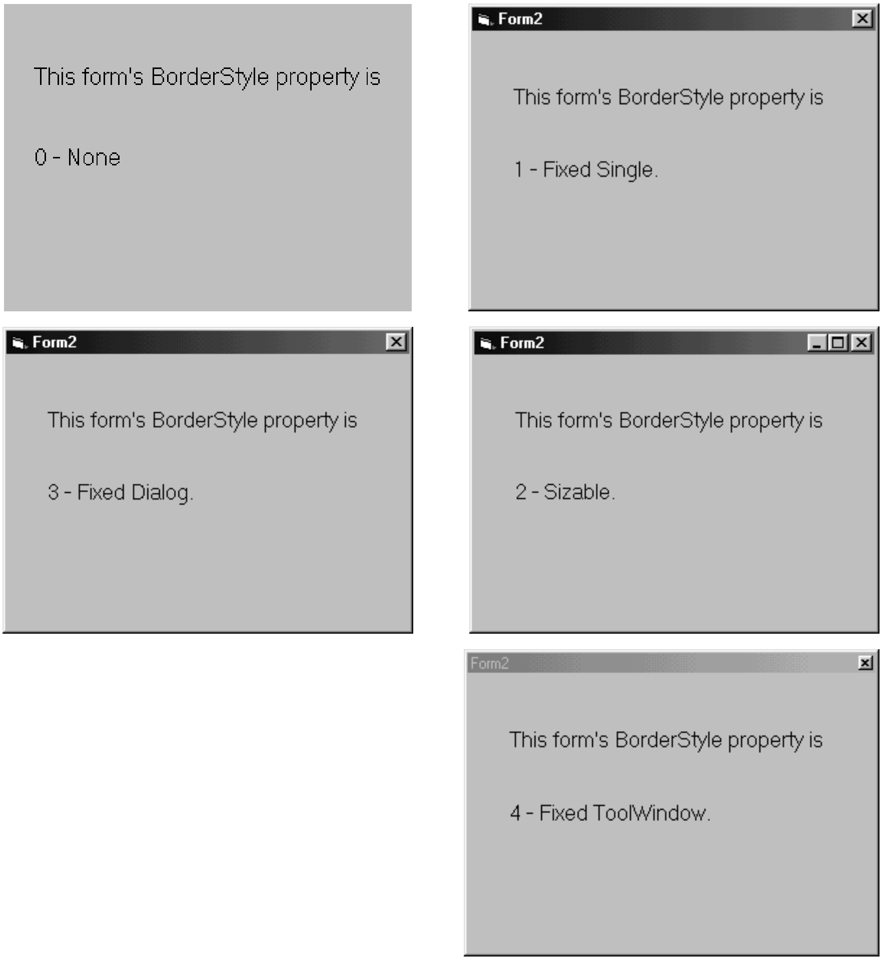
הגדרה	תוצאה
0 - None	לא מוצגת מסגרת לטופס. כמו כן לא מוצגים שורת הכותרת ולחצני הבקרה
1 - Fixed Single	מוצגת מסגרת בעלת שורה אחת. שורת הכותרת ולחצני הבקרה מופיעים. המשתמש אינו יכול לשנות את גודל הטופס
2 - Sizable	המסגרת נראית כניתנת לשינוי, שורת הכותרת ולחצני הבקרה מופיעים. המשתמש יכול לשנות את גודל הטופס על ידי גרירת אחד מגבולותיו
3 - Fixed Dialog	לטופס יש מסגרת קבועה. שורת הכותרת ולחצני הבקרה נראים. לחצני הגדל ומזער אינם מופיעים. אי אפשר לשנות את גודלו
4 - Fixed ToolWindow	לטופס יש מסגרת של שורה אחת ומציג רק את שורת הכותרת ולחצן הסגירה. שורת הכותרת מופיעה בגופן מוקטן
5 - Sizable ToolWindow	כמו Fixed ToolWindow, מלבד מסגרת שאפשר לשינוי

הערה:

העמדת המאפיין BorderStyle על ערך אשר אינו מאפשר שינוי בגודל הטופס, איננה משפיעה במהלך העיצוב. הטופס ייראה בלתי ניתן לשינוי רק בזמן פעולת התוכנית.

הגדרת ברירת המחדל במאפיין BorderStyle נותנת מסגרת המאפשרת למשתמשים לשנות את גודלו בזמן פעולת התוכנית. תוכל למצוא סוג טופס זה באפליקציות Windows טיפוסיות רבות. לעומת זאת, תוכל לשנות את המאפיין כך שהטופס ייראה כמו כל סוג חלון שאפשר לראות בתוכנית, כגון ארגזי כלים, תיבות דו-שיח, ואפשר אפילו להוריד את המסגרת לגמרי.

חלק מההגדרות שבטבלה 3.3 גורמות לתיבת הבקרה וללחצנים סגור, הגדל ומזער להופיע בשורת הכותרת של הטופס. הגדרה זו היא ברירת מחדל. עם זאת, בכל סגנון שתבחר, תוכל עדיין לקבוע באופן יחידני האם לחצנים אלו יופיעו או לא. המאפיינים ControlBox, MaxButton, MinButton מכילים כל אחד ערך True או False בעזרתם ניתן לקבוע האם האלמנט יופיע על המסך. ערך ברירת המחדל בכל אחד מהמאפיינים מכוון ל-True. אם תשנה את הערך ל-False, האלמנט לא יופיע בטופס שלך. תכונות אלה ניתנות לשינוי רק בזמן עיצוב.



תרשים 3.11: על ידי שינוי המאפיין BorderStyle תוכל לתת לטופס סגנונות הופעה רבים

המאפיין Font מאפשר לך לקבוע את הגופן הבסיסי של הטופס ואת תכונותיו. המאפיין ישפיע על כל טקסט המופיע ישירות על גבי הטופס.

הערה:

המאפיין Font של טופס הוא למעשה אובייקט בפני עצמו עם מאפיינים משלו. לדוגמה, כדי לשנות את גודל הגופן יש לכתוב Form1.Font.Size = 10 בחלון הקוד (או בחלון המיידי, לצורך העניין). פעולה זו תשנה את גודלו של הגופן ל-10 נקודות.

בנוסף לכך, שינוי המאפיין Font קובע את הגופן שישמש בכל הפקדים שיתווספו לטופס.

מאפיין נוסף, הראוי לציון מיוחד, הוא StartupPosition. כפי שאתה בוודאי מנחש, מאפיין זה שולט במיקום הטופס על המסך בעת הופעתו לראשונה. למאפיין StartupPosition ארבעה מצבים אפשריים, המופיעים בטבלה 3.4.

טבלה 3.4: הגדרות המאפיין StartupPosition

הגדרה	תוצאה
0 - Manual	המיקום הראשוני נקבע בהתאם לערכי המאפיינים Top ו-Left
1 - CenterOwner	הטופס מופיע במרכז שולחן העבודה, אלא אם כן הוא צאצא של טופס MDI. במקרה כזה הטופס ממוקם במרכז טופס האב
2 - CenterScreen	הטופס מופיע במרכז שולחן העבודה
3 - Windows Default	הטופס ממוקם על ידי Windows בהתבסס על המיקום ומספר החלונות האחרים הפתוחים באותו זמן

בעזרת המאפיין StartupPosition ניתן לגרום לטופס להופיע במרכז המסך בעת הופעתו הראשונה, עם זאת, אין זו תכונה "עקשנית", כלומר לא ניתן לשמור את הטופס במרכז במידה והמשתמש החליט לשנות את מיקומו או גודלו.

הצגת טופס

אם אתה כותב תוכנית עם טופס אחד בלבד, אין צורך לדאוג להצגת הטופס או להסתרתו. תהליך זה מבוצע אוטומטית עבורך ברגע שהתוכנית מתחילה או מפסיקה. טופס יחיד זה נקרא **אובייקט ההפעלה** (Startup Object) או טופס ההפעלה. כאשר אתה מפעיל את התוכנית, Visual Basic טוענת את טופס ההפעלה לזיכרון ומציגה אותו. כל עוד טופס זה טעון בזיכרון, התוכנית תמשיך לפעול ולהגיב לאירועים. התוכנית תפסיק רק כאשר תלחץ על לחצן הסגירה (או תפעיל את הפקודה End).

הערה:

את טופס ההפעלה אפשר לבחור מתיבת הדו-שיח של מאפייני הפרויקט. בנוסף לכך אפשר לגרום לתוכנית להתחיל מתת-השיגרה Main בקוד במקום להתחיל בטופס.

לעומת זאת, אם בתוכנית יש מספר טפסים, כמו במרבית התוכניות, עליך להבין כיצד לנהל אותם. מצבו של טופס נקבע ב- Visual Basic על ידי המשפטים Load ו-Unload וכן על ידי השיטות Show ו-Hide.

המשפט Load טוען את הטופס לזיכרון אך לא מציג אותו. שורת הקוד הבאה מראה כיצד להשתמש במשפט:

```
Load frmApplication
```

על ידי שימוש במשפט זה, אתה טוען את הקובץ לזיכרון במפורש. לעומת זאת, אם תפנה לכל אחד מהאובייקטים שבו - פקדים, שיטות או מאפיינים שלו, הטופס יוטען באופן אוטומטי לזיכרון. בגלל שטעינת הטופס היא אוטומטית, השימוש במשפט Load אינו ממש נחוץ. למרות זאת, חשוב להיות מודעים לזמן טעינת הטופס מכיון שכאשר הטופס מוטען הוא מפעיל את הקוד שמופיע תחת האירוע Form_Load.

בכדי להציג כל טופס אחר מלבד טופס ההפעלה, עליך להשתמש בשיטה Show. השיטה Show עובדת ללא קשר אם הטופס הוטען לזיכרון קודם לכן או לא. במידה והטופס לא הוטען, Show מטעינה את הטופס לזיכרון ומייד מציגה אותו. יש להשתמש בשיטה Show באופן הבא:

```
frmApplication.Show
```

השיטה Show מכילה גם ארגומנט אשר קובע האם הטופס יוצג במצב **מודאלי** (Modal) או **לא מודאלי** (Modeless). אם הטופס מוצג במצב מודאלי, השליטה בתוכנית לא חוזרת לשיגרה שהפעילה את השיטה Show עד שהטופס המודאלי נסגר. חשוב על מצב זה כעל השהיית קוד התוכנית כל עוד הטופס המודאלי מוצג. דוגמה לטופס מודאלי הוא מסך הכיבוי של Windows 95. אין אפשרות להעביר את המוקד לאף מסך אחר כל עוד מסך הכיבוי מוצג.

אם הטופס מוצג במצב לא מודאלי, המשתמשים בתוכנית יוכלו לעבור כרצונם בין הטופס הנוכחי לבין טפסים אחרים בתוכנית. המשפט שהוצג קודם לכן הציג טופס במצב לא מודאלי. כדי לפתוח אותו במצב מודאלי, צריך פשוט לשנות את ארגומנט השיטה Show ל-vbModal, כפי שמתואר לעיל:

```
frmApplication.Show vbModal
```

הערה:

מצב מודאלי משמש לרוב כאשר תרצה שהמשתמש יסיים את הפעולות על הטופס לפני שיעבור לכל חלק אחר בתוכנית. אם מופיעה, לדוגמה, הודעת שגיאה קריטית, אתה בוודאי לא תרצה שהמשתמש יתעלם ממנה ויעבור לטופס אחר.

לאחר שהטופס מוצג, יש שתי שיטות תכנות להיפטר ממנו. השיטה Hide מסירה את הטופס מהמסך אך לא מהזיכרון. השתמש ב-Hide כשאתה רוצה להעלים את הטופס באופן זמני אך עדיין צריך את האינפורמציה שבתוכו:

```
frmApplication.Hide
sUserName = frmApplication.txtUserName.Text
```

בדוגמה זו, שורת הקוד השנייה יכולה לגשת למידע שבפקד ב-frmApplication, למרות שטופס זה לא נראה על המסך. השיטה Hide משאירה את הטופס בזיכרון.

אם סיימת עם טופס ועם המידע הכלול בו, אפשר להסיר אותו גם מהמסך וגם מהזיכרון על ידי שימוש במשפט Unload. המשפט Unload משתמש באותו תחביר כמו המשפט Load, כמוצג לעיל:

```
Unload frmApplication
```

טיפ:

אם אתה משתמש בפקודה Unload מתוך הטופס אותו אתה מתכוון להסיר, אפשר להשתמש במילת המפתח Me כדי להגדיר את הטופס. שימוש בגישה זו ימנע טעויות אם תשנה את שם הטופס מאוחר יותר. במקרה כזה המשפט ייראה כך:

```
Unload Me
```

שיפור הביצועים בתוכנית על ידי שימוש ב-Load

מכיון שהשיטה Show טוענת טופס אוטומטית לזיכרון, אינך צריך בדרך כלל להשתמש במשפט Load בתוכנית כלל. למרות זאת, טפסים מסוימים, המכילים מספר פקדים רב, עלולים להיטען באיטיות רבה. דרך אחת להתגבר על תופעה זו היא לטעון את הטופס לזיכרון על ידי שימוש במשפט Load ברגע שהתוכנית מופעלת. כל הפעלה של שיטת Hide או Show לטופס תגיב באופן מהיר יותר ברגע שהטופס כבר נמצא בזיכרון. אם תבחר להשתמש בטריק זה, היזהר משני דברים: ראשית, אל תשכח לבצע Unload על הטופס לאחר שתסיים להשתמש בו או בסוף התוכנית. שנית, הייה מודע לאפשרות של מגבלות זיכרון. אם תטעין מספר רב מדי של טפסים לזיכרון בבת אחת, אתה עלול להרגיש בירידה תלולה בביצועי התוכנית.

טעינת טפסים לזיכרון אמנם מגדילה את כמות הזמן העובר עד לטעינת התוכנית לזיכרון, אך מצד שני חוסכת זמן בכל פעם שברצונך להציג טפסים. אם אתה מציג טופס מסוים פעם אחת בלבד בתוכנית, אין שום חיסכון בהטענה מוקדמת שלו. לעומת זאת, אם טופס מופיע מספר פעמים, הטענתו המוקדמת תחסוך זמן, בדרך כלל. בנוסף, עליך להיות מודע לכך שהמשתמשים בדרך כלל סבלניים יותר לזמן הטעינה הראשוני של התוכנית (במיוחד אם הם עסוקים בהתבוננות במסך הפתיחה), מאשר כשהם מנסים לבצע פעולה.

מכאן...

פרק זה הציג בפניך את עולמם של הטפסים והפקדים. התחלת לראות כיצד להשתמש בטפסים ופקדים כאבני הבניין של ממשק המשתמש. למדת כיצד לנהל את צורת הופעתם של טפסים ופקדים וכיצד לשנות את מאפייניהם הן בזמן התכנון והן בזמן ההרצה. המושגים הבסיסיים שנדונו כאן ילוו אותך גם בפרקים הבאים, אשר יכללו הסברים על רבים מהפקדים בהם תוכל להסתייע בעת בניית אפליקציות ב- Visual Basic.

- למד עוד על פקדים סטנדרטיים על ידי קריאת פרק 4 **שימוש בפקדי ברירת המחדל של Visual Basic**.
- פרק 5 **תגובה באמצעות שגרות אירוע**, יעמיק את היכרותך עם שגרות אירוע.
- תוכל לגלות עוד אודות Visual Basic בפרק 8 **שימוש במשתנים לאחסון מידע**, ופרק 11 **ניהול הפרויקט: תת-שגרות, פונקציות וריבוי טפסים**.

4

שימוש בפקדי ברירת המחדל של Visual Basic

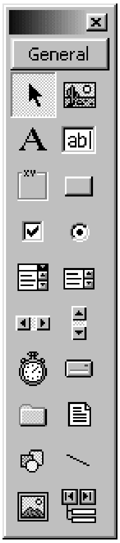
בפרק זה?

- מבוא לפקדים פנימיים
- עבודה עם טקסט
- פקדים לבחירת אפשרויות
- פקדים למטרות מיוחדות
- עבודה עם פקדים מרובים בשלב העיצוב
- עבודה עם אוסף הפקדים

בשלב זה אמור הקורא לדעת מהו פקד וכיצד משתמשים בפקדים בתוכניות Visual Basic. דרך טובה ללמוד יותר על Visual Basic היא לחקור כל אחד מהפקדים הזמינים. פקדים נדונים כמעט בכל פרק בספר זה. שלושת הפרקים הבאים מתמקדים בנושא השימוש בפקדים מסוימים.

מבוא לפקדים פנימיים

Visual Basic מגיעה עם פקדים המאפשרים ביצוע סוגים רבים של משימות תכנות. כמה מפקדים אלה מופיעים אוטומטית בארגז הכלים כאשר Visual Basic נפתחת. פקדים תקניים אלה נקראים פקדים **פנימיים** (Intrinsic). פקדים אלה, המוצגים בתרשים 4.1, כוללים כמה פקדים מאוד כלליים, שסביר כי ישמשו כל מפתח Visual Basic. אלה מרוכזים בטבלה 4.1.



תרשים 4.1: ארגז הכלים מציג כלים המייצגים את הפקדים המובנים בשפת Visual Basic

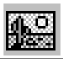
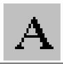
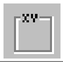
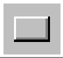
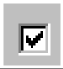
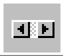
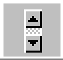
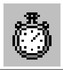
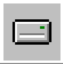
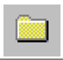
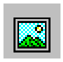
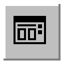
הערה:

הפקדים המופיעים בתרשים 4.1 ומתוארים בטבלה 4.1 אינם היחידים הנכללים ב- Visual Basic. להוספת פקדים אחרים, כגון פקד Microsoft CommonDialog או Microsoft FlexGrid לארגז הכלים, יש להשתמש בתיבת הדו-שיח **Components** על ידי בחירת **Project**, **Components**, או ללחוץ לחיצה ימנית באזור ריק של ארגז הכלים, ולבחור מהתפריט **Components**.

רבים מהפקדים הרשומים בטבלה 4.1 יידונו בקטעים הבאים. פקדים אחרים מתוארים בפרקים העוסקים בנושא תואם למטרותיהם. הפקדים המתוארים בפרקים אחרים הם:

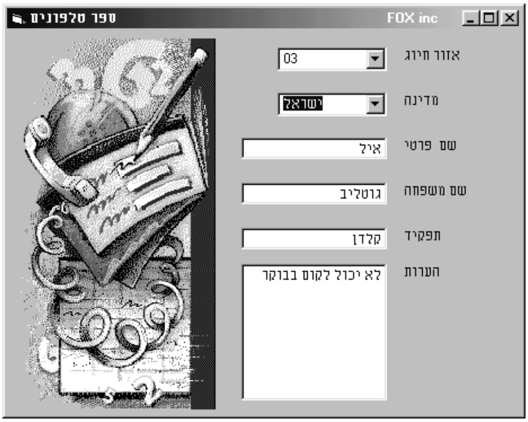
- פקדי PictureBox, Image, Shape, Line, נדונים בפרק 19 **שימוש במרכיבי התכנון הוויזואלי**.
- הפקד Data מטופל בפרק 25 **פקד הנתונים ופקדי איגוד נתונים**.
- הפקד OLE נדון בפרק 22 **שימוש ב-OLE לשליטה על יישומים אחרים**.

טבלה 4.1: פקדים הנכללים בארגז הכלים של Visual Basic

תמונה	שם הפקד	שימוש
	PictureBox	מציג גרפיקה. משמש גם להכלת פקדים אחרים
	Label	מציג טקסט שאינו ניתן לעריכה בידי המשתמש
	TextBox	מציג טקסט. מאפשר לשנות את הטקסט
	Frame	מכולה לפקדים אחרים. מאפשר קיבוץ פקדים
	CommandButton	מאפשר התחלת פעולה בלחיצה על לחצן
	CheckBox	מאפשר בחירה מסוג נכון/לא נכון
	OptionButton	מאפשר בחירת אפשרות אחת מקבוצת פריטים
	ComboBox	תיבת טקסט נפתחת לרשימת ערכים
	ListBox	מאפשר בחירה מתוך רשימת פריטים
	Horizontal ScrollBar	פס גלילה אופקי
	Vertical ScrollBar	פס גלילה אנכי
	Timer	מאפשר לתוכנית לבצע פעולות על בסיס מתוזמן
	Drive List Box	מאפשר בחירת כונן דיסקים
	Directory List	מאפשר בחירת ספריה או תיקיה
	File List Box	מאפשר בחירת קובץ
	Shape	מציג צורה בטופס
	Line	מציג קו בטופס
	Image	דומה ל-PictureBox. משתמש בפחות משאבי מחשב, תומך בפחות מאפיינים, אירועים ושיטות
	Data Control	מספק ממשק בין התוכנית למקור נתונים
	OLE	מספק קשר בין התוכנית לשרת OLE
	Common Dialog	שימוש בתיבות דו שיח תקניות של Windows, לשליפת מידע כגון שמות קובץ, גופנים וצבעים

עבודה עם טקסט

בפרק 2 **יצירת התוכנית הראשונה**, הוצגה בניית תוכנית פשוטה, העושה שימוש בפקדים TextBook, Label. מומלץ לקורא, אם לא עבר על פרק זה, לעשות זאת לפני שימשיך. הדוגמאות לימדו שפקדי תיבת הטקסט והתווית מיועדים לעבודה עם טקסט. המונח **טקסט** (Text) אינו מתייחס כאן רק לפסקאות או משפטים כמו אלה המטופלים במעבד תמלילים. בטיפול בטקסט בתוכנית, דרושה לעיתים גם הצגה או שליפה של מילה בודדת, מספר, או אפילו תאריך. הקורא יכול להבחין גם בהבדל העיקרי שבין תיבת טקסט לתווית: תיבת הטקסט יכולה גם להציג וגם לקלוט קלט טקסט (בזמן ריצה), בעוד שפקד התווית מיועד רק להצגתו. תרשים 4.2 מראה דוגמאות שונות לשימוש בפקדי TextBox ופקדי Label.



תרשים 4.2: פקדי Label מזהים את מטרות חלקי הטופס, בעוד תיבות טקסט מאפשרות הזנת נתונים או מציגות מידע

הצגת טקסט בעזרת הפקד Label

מטרת פקד Label היא להציג טקסט. לעיתים קרובות מאוד הוא משמש להצגת פריטים בטופס למשתמשים. הדרך הפשוטה ביותר להשתמש בפקד התווית היא להציב אותו לאחר שדה קלט, ולקבוע את המאפיין Caption שלו. ביחס לתרשים 4.2, יש לשים לב שפקדי Label משמשים להודעה למשתמש איזה סוג של מידע יש להכניס לתיבות הטקסט.

למרות שהפקד Label אינו מאפשר הזנה ישירה של טקסט, הוא כולל אירועים ומאפיינים המאפשרים למפתח לתפעל את הטקסט ישירות מהקוד. כדי לנסות זאת, בנה פרויקט EXE תקני חדש והצב בטופס פקד Label. הצב את הקוד הבא באירוע click:

```
Private Sub Label1_Click()
      Label1 = "The time is: " & Time$
End Sub
```

הרץ את התוכנית. בכל פעם שתלחץ על התווית בעזרת העכבר, תציג התווית את השעה. בנוסף, שים לב שבקוד לעיל לא הוזכר בפירוש המאפיין Caption, מכיון שזה מאפיין המחדל של הפקד Label. כזכור, משתמשים בסימון הנקודה לציון המאפיין של אובייקט, כמו בשורת הקוד הבאה:

```
Label1.Caption = "The time is " & Time$
```

אולם, פקד Label אינו מוגבל להצגת כמויות קטנות של טקסט בלבד. למעשה ניתן להשתמש בפקד Label להצגת מספר שורות או אפילו פסקאות מידע. אם רוצים להציג כמות מידע גדולה בפקד זה, יש להקדיש תשומת לב מיוחדת למאפיין AutoSize.

המאפיין AutoSize

אם ידוע איזה טקסט עומד להיות מוצג, ניתן לקבוע את גודל התווית שתתאים לקליטת הטקסט. אולם, אם יוצגו בתווית טקסטים שונים (למשל ביישומי בסיס נתונים), כדאי לאפשר להתאים את התווית לתכולתה בזמן נתון. המאפיין AutoSize קובע אם גודל הפקד יותאם אוטומטית לטקסט המוצג. כאשר AutoSize הוא במצב False (מצב המחדל), גודל התווית נשאר קבוע, בלי קשר לכיתוב המוצג. אם הכיתוב ארוך מדי לתווית, חלק ממנו לא ייראה, מכיון שאין הוא גולש לשורה הבאה.

טיפ:

בקביעת המאפיין Caption, ניתן לאלץ שורה חדשה על ידי הכללת שילוב של פעולת החזרת הגררה והזנת שורה. הטכניקה, כמו המינוח, היא שריד מתקופת מכונות הכתיבה הידניות. כשהמשתמש היה מגיע לקצה השורה, היה עליו לקדם את הדף ידנית שורה אחת (הזנת שורה), ולהחזיר את הגררה לתחילת השורה (החזרת גררה). Visual Basic מאפשרת להכניס שילוב החזרת גררה/הזנת שורה על ידי הכללת תווי ASCII מספר 13 ו-10, במקום שבו אמורה להיפתח שורה חדשה. Visual Basic מספקת קבוע מוגדר מראש vbCrLf, לעזרה במקרים אלה:

```
Label1.caption = "First Line" & vbCrLf & "Second Line"
```

השליטה בהופעת הטקסט

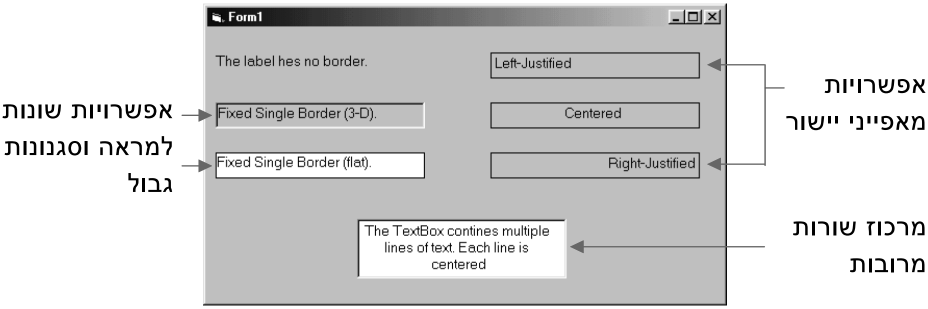
מאפיין Caption מכיל טקסט אמור להצגה, מאפיינים אחרים מפקחים על הופעת הטקסט. AutoSize תואר בפיסקה הקודמת. מאפיינים נוספים מתוארים להלן:

- Alignment - מבקר את יישור הטקסט (שמאלי, ימני או ממורכז).
- Appearance - גורם לטקסט להיראות שטוח או תלת-מימדי.
- BorderStyle - קובע אם לפקד Label יש גבול.
- Font - הופך את הטקסט למודגש, בעל קו תחתי או נטוי, או משנה את הגופן.
- ForeColor ו-BackColor - קובעים את צבע הטקסט וצבע הרקע.
- UseMnemonic - קובע אם התו & במאפיין Caption מטופל כמציין קוד גישה.

הערה:

בקביעת BorderStyle כ- Fixed Single נראה פקד התוויות כתיבת טקסט בלתי ניתנת לעריכה.

להדגשת טקסט בתווית ובפקדים אחרים, ניתן לקבוע מאפיינים אלה כך שיאפשרו כל מספר של אפקטים חזותיים. התוצאה של קביעת אחדים ממאפיינים אלה מוצגת בתרשים 4.3.



תרשים 4.3: המאפיינים Alignment, Appearance וכן BorderStyle יכולים לשנות את צורת פקד Label

הערה:

המאפיין Alignment משפיע על הופעת הטקסט גם כאשר התווית מציגה אותו ביותר משורה אחת. הפקד מיישר כל שורה לפי קביעת Alignment. (ראה תרשים 4.3).

הזנת טקסט בתיבת טקסט

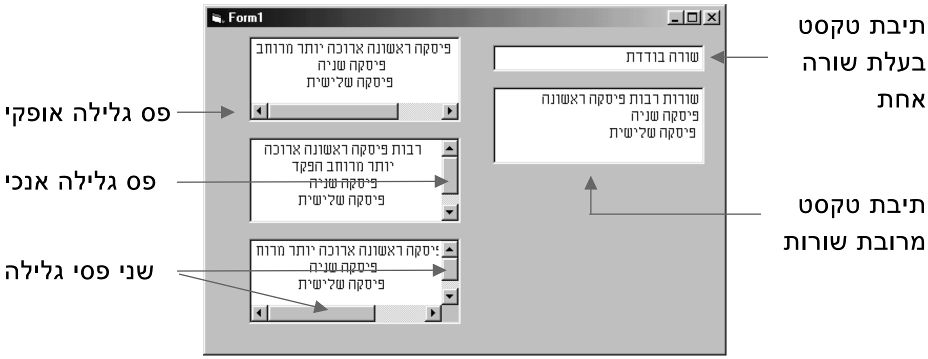
מכיון שחלק גדול מפעולת התוכנית כולל שליפה, עיבוד והצגה של טקסט, ניתן להניח שסוס העבודה העיקרי של תוכנית רבות הוא הפקד TextBox. תיבת הטקסט מאפשרת הצגת טקסט. אך בנוסף היא מאפשרת למשתמשי התוכנית גישה נוחה להכנסת טקסט ולעריכתו, ולתוכנית היא מאפשרת לשלוף את המידע שהוזן.

הטיפול ביותר משורת טקסט אחת

ברוב המקרים משמשת תיבת הטקסט לטיפול בקטע מידע אחד, למשל שם או כתובת. אך תיבת הטקסט יכולה לטפל גם באלפי תווי טקסט. תכולת פקד TextBox מאוחסנת במאפיין Text שלו – המאפיין העיקרי שאיתו "משוחחת" התוכנה. ניתן גם להגביל את מספר התווים שמשתמש יכול להזין על ידי המאפיין MaxLength.

ברירת המחדל לתיבת טקסט היא קליטת שורת טקסט אחת. כמות מידע זו מספיקה ברוב המקרים, אך לעיתים נדרשת התיבה לטפל בכמות טקסט גדולה יותר. שני מאפיינים שימושיים במקרה של טיפול בכמויות טקסט גדולות בתיבת טקסט הם: MultiLine ו-ScrollBar.

המאפיין MultiLine קובע אם המידע בתיבת הטקסט מוצג בשורה בודדת, או גולש ונגלל לשורות אחדות. אם מאפיין MultiLine נקבע ל-True, המידע מוצג בכמה שורות וגלישת מילים מטופלת באופן אוטומטי. המשתמש יכול להקיש ENTER כדי לאלץ שורה חדשה. המאפיין ScrollBar קובע אם פסי גלילה יוצגו בתיבת הטקסט, ואם כן - מאיזה טיפוס יהיו (None, Horizontal, Vertical או Both). פסי הגלילה שימושיים במקרים בהם מאוחסן במאפיין Text יותר טקסט מהתואם את תיבת הטקסט. המאפיין ScrollBar משפיע על תיבת הטקסט רק אם המאפיין MultiLine נקבע כ-True. תרשים 4.4 מראה את השפעת מאפייני MultiLine וכן ScrollBar.



תרשים 4.4: ניתן להשתמש בתיבת טקסט להזנת שורות בודדות או פסקאות שלמות



טיפ:

כאשר משתמש עובר לתיבת טקסט בהקשה על מקש הטאב או בלחיצת העכבר בתוכה, פעולה המוכרת כקבלת **מיקוד** (Focus), מקובל שהטקסט נבחר (או מואר). למרות שפעולה זו אינה מתבצעת אוטומטית, אפשר לעשות זאת בקלות יחסית. הוסף את שורות הקוד הבאות בשגרת האירוע של תיבת הטקסט, תוך החלפת Text1 בשם תיבת הטקסט שלך:

```
Text1.SelStart = 0
Text1.SelLength = Len(Text1.Text)
```

אימות קלט

בקליטת טקסט חופשי ממשתמשים, יש לעיתים קרובות צורך באימות כלשהו של הנתונים לפני שממשיכים. למשל, במקרה תיבת טקסט המקבלת מידע שהוא מספר טלפון, ניתן להשתמש בקוד לבדיקת אורך מספר הטלפון או אפילו לוודא שהמשתמש הזין בכלל מידע.

מאפיין חדש בגרסה Visual Basic 6 הוא אירוע תיבת הטקסט Validate. שגרת אירוע זו יכולה לכלול קוד המאמת את המידע בפקד הקשור אליו. הוא מופעל על ידי פקד אחר, שהמאפיין CausesValidation שלו נקבע כ-True. למשל, שורת הקוד הבאה מציגה תיבת הודעה אם תיבת הטקסט איננה מכילה נתונים נומריים:

```
If Not IsNumeric(Text) Then MsgBox "Please enter a number"
```

לניסוי האירוע בנה פרויקט EXE תקני חדש. מקם בטופס לחצן פקודה ותיבת טקסט, והזן את שורת הקוד לעיל באירוע Validate של תיבת הטקסט. בהקשת טאב לצורך יציאה מתיבת הטקסט, תוצג ההודעה, אלא אם כן הוזן לתיבה מספר. קביעת המאפיין CausesValidation כ-False תפסיק את אירוע האימות.

פקדים לבחירת אפשרויות

בקטע הקודם למדת איך לקלוט מידע ממשתמשים דרך תיבת הטקסט. גישה זו מתאימה למקרים רבים של איסוף מידע. אולם מה קורה אם דרושה רק פיסת מידע קטנה, כגון "היש בבעלותך מכונית?" או "מה מצבך המשפחתי?" במקרים כאלה עומדות לרשות המשתמש רק שתיים או לכל היותר מספר קבוע של תשובות אפשריות. אם למשל התוכנית בנויה לקלוט רק את המילים **כן ולא**, תיווצר בעיה אם משתמש מקליד "אולי", או טועה בהקלדה.

ניתן למנוע את הבעיה ולהפוך את התוכנית נוחה יותר לשימוש, אם משתמשים בפקדים להצגה וקליטה של אפשרות בחירה. בקטעים הבאים ייבחנו פקדים אחדים המאפשרים ברירה בין אפשרויות. ניתן להציע ברירה כזאת בשימוש בתיבות סימון, לחצני אפשרויות, רשימות ותיבות משולבות:

- **תיבת סימון** (Check Box) - מפעילה או מפסיקה אפשרות אחת או יותר.
- **לחצן אפשרויות** (Option Button) - בוחר אפשרות אחת מתוך רשימה.
- **תיבת רשימה** (List Box) - מציגה רשימת פריטים המוגדרים על ידי המשתמש.
- **תיבה משולבת** (Combo Box) - תיבת טקסט עם רשימה נפתחת.

לחצן הפקודה

הפקד **CommandButton** הוא בעל חשיבות כמעט בכל יישום. הוא מאפשר בדרך כלל אתחול פעולות בלחיצה על לחצן. ניתן להתקין פקד CommandButton על ידי הצבת הלחצן על הטופס וקביעת המאפיין Caption שלו בתור הטקסט שאותו מבקשים להציג על פני הלחצן. להפעלת הלחצן יש להציב רק קוד בשגרת האירוע Click שלו. כמו בכל שגרת אירוע, יכול קוד זה לכלול כל מספר של משפטי תכנות תקפים של Visual Basic.

למרות שמשתמשים מעדיפים לעיתים קרובות ללחוץ על לחצני פקודה באמצעות העכבר, ישנם המעדיפים גישה לפקודה מן המקלדת. זהו לעיתים קרובות המצב בתוכניות עתירות הזנת נתונים. כדי לספק משתמשים אלה, ניתן לגרום לתוכנית להפעיל אירוע של לחצן פקודה בלחיצה על מקשים מסוימים במקלדת. עושים זאת על ידי הקצאת מקש גישה ללחצן הפקודה. כשהוגדר מקש גישה, ניתן להפעיל בהקשה עליו, תוך החזקת מקש Alt, את האירוע Click של הפקד CommandButton.

הקצאת מקש גישה מתבצעת בקביעת מאפיין Caption של הפקד CommandButton. יש להציב את התו & לפני אות המקש הנבחר. למשל, כדי לאפשר למשתמש להדפיס בהקשת Alt+P, קובעים את מאפיין Caption כ-&Print. התו & אינו מופיע על הלחצן, אך אות מקש הגישה מודגשת בקו תחתון. על לחצן הפקודה יופיע הכיתוב <u>P</u>rint.

הערה:

אם דרוש מסיבה כלשהי להציג בכיתוב לחצן הפקודה את התו &, פשוט הצב אותו פעמיים במאפיין Caption. למשל: Save && Exit יוצר את הכיתוב Save & Exit.

ניתן ליעד לחצן פקודה אחד על כל טופס כ**לחצן מחדל** (Default Button). לחיצה על Enter כשפקד כלשהו נמצא במיקוד (למעט לחצן פקודה אחר או תיבת טקסט שבה האפיון Multiple הוא True), תגרום להפעלת לחצן המחדל. פעולה זו מפעילה את האירוע Click של לחצן המחדל, בדיוק כאילו לחץ המשתמש על לחצן המחדל. לקביעת לחצן כלחצן מחדל יש לקבוע את המאפיין Default שלו כ-True. רק לחצן אחד בטופס יכול להיות לחצן מחדל.

ניתן גם לייעד לחצן אחד כ**לחצן ביטול** (Cancel Button), הדומה ללחצן המחדל, אך פועל עם מקש Esc. להסבת לחצן פקודה ללחצן ביטול, קבע את המאפיין Cancel שלו כ-True. כמו לגבי לחצן המחדל, רק לחצן אחד בטופס יכול להיות לחצן ביטול. עם קביעת ערך המאפיין Default או Cancel של לחצן אחד כ-True, מוסב מאפיין זה בלחצנים האחרים ל-False.

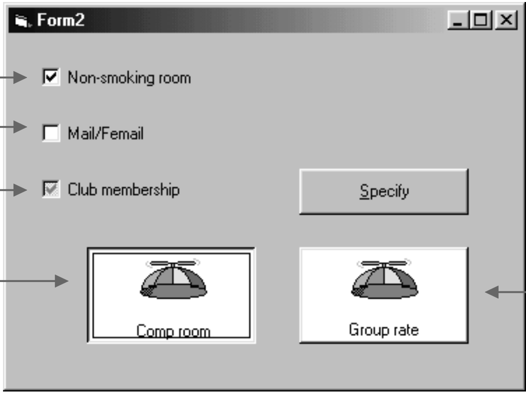
תיבות סימון

פקד **תיבת הסימון** (Check Box) של Visual basic משמש לקבלת תשובת "כן" או "לא" מהמשתמש. פקד זה פועל כמפסק חשמלי. או שהוא מופעל, או שהוא מושבת. אין מצבי ביניים. כשתיבת סימון מופעלת, מוצג הסימן ✓ בתיבה. הסימן מציין שהתשובה לשאלה הקשורה לתיבת הסימון היא "כן". כשהתיבה ריקה ואין בה סימון, התשובה היא "לא". לתיבת הסימון מצב אפשרי נוסף, מצב **אפור**, המוצג כסימן ✓ על רקע אפור. מצב זה מצביע בדרך כלל על ביצוע בחירה חלקית (נבחרו רק כמה מתוך תת-אפשרויות הבחירה, לא כולן). המשתמש יכול לבטל את הבחירה החלקית, ואז נמחק הסימון האפור. אם הוא בוחר בתיבה מחדש, כל אפשרויותיה נבחרות. תרשים 4.5 מציג את שלושת המצבים האפשריים של תיבת הסימון.

הערה:

סימן ✓ או תבנית התיבה הריקה אופייניים לצורה התקנית של תיבת הסימון. אם קובעים את ערך המאפיין Style כ- 1 - Graphical, משמשות תמונות לסמן תיבת סימון מלאה או ריקה. תיבות סימון גרפיות מודגמות גם הן בתרשים 4.5.

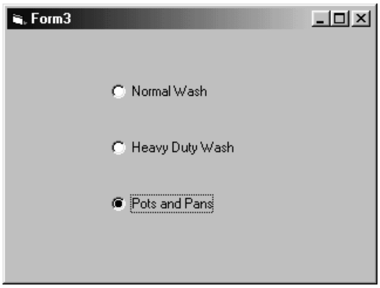
תרשים 4.5 מדגים, בנוסף לתיבות הסימון התקניות, גם תיבות סימון גרפיות. שני מאפיינים, Picture ו-DownPicture, קובעים את התמונות המוצגות בתיבות הגרפיות.



תרשים 4.5: תיבת הסימון יכולה לציין "כן" או "לא" כתגובה לשאלה

לחצני אפשרויות

לחצני אפשרויות (Option Button), הנקראים גם **לחצני רדיו** (Radio Button), דומים ללחצנים במכשיד רדיו של מכונית: הם קיימים בקבוצות, וניתן "לבחור" רק אחד מהם בכל זמן נתון. הם שימושיים להצגת רשימה אפשרויות קבועה, מהן ניתן לבחור בו-זמנית רק אפשרות אחת. לבדיקת הנושא הצב כמה לחצני אפשרויות על טופס. בתחילה יש לכל לחצן ערך False. אם משנים ערך של אחד מהלחצנים ל-True, ולאחר מכן משנים ערך של לחצן אחר ל-True, הלחצן הקודם שהיה True יהפוך אוטומטית ל-False, כמודגם בתרשים 4.6.



תרשים 4.6: לחצני אפשרויות יכולים לשמש להצגת רשימה הקבועה של אפשרויות בחירה

בקוד ניתן להשתמש בלחצני אפשרויות באחת משתי דרכים:

- השתמש באירוע Click אם בכוונתך לבצע פעולה כשהמשתמש בוחר באפשרות. שיטה זו שימושית כשמשתמשים בלחצני פיקוד הנמצאים במערך פקדים, כמודגם בדוגמה הבאה:

```
Private Sub optWash_Click(Index As Integer)
    Select Case Index
        Case 0
            MsgBox "You selected: Normal"
        Case 1
            MsgBox "you selected: Heavy Duty"
        Case 2
            MsgBox "You selected: Pots and Pans"
    End Select
End Sub
```

* אל תכתוב שום קוד באירוע לחצני האפשרות. במקום זאת עשה שימוש בשורת If לבדיקת מצבם, כמודגם להלן:

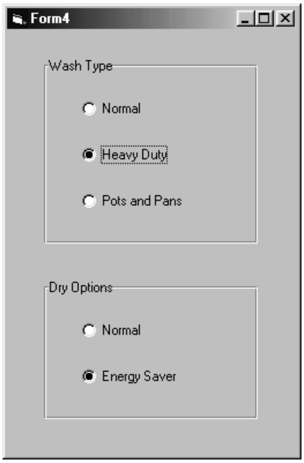
```
Private sub cmdStartWash_Click( )

    If optHeavy = True Then
        DoHeavyWash
    Else
        DoNormalWash
    End If

End Sub
```

השיטה השנייה שימושית אם אינך מעוניין שמשהו יקרה מייד לאחר שהמשתמש בחר באפשרות.

האם ניתן לבחור יותר מלחצן אפשרויות אחד בטופס? התשובה חיובית, אך מחייבת חלוקת לחצני האפשרויות לקבוצות, תוך שימוש בפקד המשמש כ**מכולה** (Container). השימוש במכולות, כמו למשל פקד ה**מסגרת** (Frame), שיידון בהמשך הפרק, מאפשר ריכוז לחצני האפשרויות בקבוצות, כמודגם בתרשים 4.7.



תרשים 4.7: ליצירת מספר קבוצות של לחצני אפשרויות, יש להציבן במכולה, כגון מסגרת או תיבת תמונה

תיבת הרשימה

תיבת רשימה (List Box) משמשת להצגת רשימת אפשרויות. תרשים 4.8 מציג רשימה פשוטה, המשמשת לבחירת מדינה, לצורך הדפסה בתווית כתובת. הרשימה מציגה את כל המרכיבים של תיבת הסימון.

חלקי המפתח של תיבת סימון הם:

* **רשימת פריטים**. רשימת הפריטים מתוכה אמור המשתמש לבחור. פריטים אלה נוספים לרשימה בסביבת התכנון, או על ידי התוכנית תוך כדי הרצתה.
* **פריט נבחר**. הפריט שנבחר על ידי המשתמש. בהתאם לסוג הרשימה המוצג, מסומן הפריט הנבחר בהארה או על ידי סימן ✓ בתיבת הסימון ליד הפריט.
* **פס גלילה**. חלק זה מציין שברשימה כלולים פריטים נוספים שמחוסר מקום אינם מוצגים, ומספקת למשתמש אמצעי נוח להצגת פריטים נוספים אלה.

תרשים 4.8: תיבת רשימה פשוטה מכילה סדרת אפשרויות שמהן יכול המשתמש לבחור את הרצויה לו



מבחינת המשתמש, השימוש בתיבת רשימה דומה לבחירת ערוצים בטלוויזיה בכבלים. חברת הכבלים מחליטה אילו ערוצים לכלול ברשימת הערוצים לבחירה. הלקוח יכול לבחור כל ערוץ הנכלל ברשימה, אך איננו יכול להוסיף אליה, אם אין הוא מרוצה מהאפשרויות שהועמדו לרשותו. בתיבת הרשימה נקבעת על ידי המפתח רשימת האפשרויות לבחירה. המשתמשים יכולים לבחור רק מתוך אותם פריטים שהמפתח העמיד לרשותם.

כשמציירים לראשונה תיבת רשימה על טופס, מוצגים רק גבולות התיבה והטקסט List1 (שם התיבה). אין בתיבה פס גלילה וכמובן ששום רשימה אינה מופיעה שם. פס גלילה אנכי נוסף אוטומטית לתיבת הרשימה, כשמוזנות אליה יותר אפשרויות משניתן להציג בה-בעת. יש לשים לב שהתיבה אינה כוללת פס גלילה אופקי, גם אם הפריטים ברשימה ארוכים מרוחב התיבה. חשוב לוודא כי רוחבה מתאים להצגת כל הרישומים.

שימוש בסיסי בתיבת הרשימה

הדרך הפשוטה לשלוט באפשרויות הזמינות בתיבת הרשימה, היא להשתמש בשיטה AddItem להוסיף לה פריטים. נדרש רק לציין את הטקסט שמבקשים לכלול בתיבה. כדי לנסות זאת, פתח פרויקט EXE תקני חדש. צייר בטופס תיבת רשימה, והוסף את שורות הקוד הבאות לאירוע Load של הטופס:

```
Private Sub Form_Load()
	Dim i As Integer

	For i = 1 To 100
		List1.AddItem "This is item " & i
	Next i

End Sub
```

הערה:
למרות שדוגמה זו משתמשת בקוד להוספת פריטים, ניתן בשלב התכנון גם ליצור תיבה על ידי הדפסת הטקסט האמור להופיע בתיבת הרשימה, במאפיין List שלה. כל שורה במאפיין מתייחסת לאפשרות בחירה המוצגת למשתמש. לאחר הוספת פריט לרשימה יש להקיש Ctrl+Enter למעבר לשורה הבאה ברשימה.

הרץ את הפרויקט, ואז תראה שורות טקסט מופיעות בתיבת הרשימה. למחיקת כל הפריטים מתיבת רשימה, השתמש בשיטת Clear, כמו בדוגמה הבאה:

```
List1.Clear
```

הפריטים בתיבת הרשימה מאוחסנים במערך הניתן לגישה דרך המאפיין List. מערך הרשימה מתחיל באינדקס אפס, ולכן ניתן להשתמש בשורה הבאה לצורך הכנסת הפריט הראשון ברשימה:

```
Print List1.List(0)
```

ניתן לבחון שורה זו על ידי הכנסת תוכנית הדוגמה למצב עצירה (הקש Ctrl+Break), והזנת שורת Print בחלון המופיע מייד לאחר מכן.

עוד שני מאפיינים קריטיים לשימוש בתיבת הרשימה הם: ListCount המייצג את מספר הפריטים ברשימה, ו-ListIndex, מייצג את הפריט הנבחר.

אזהרה:

כשמשתמשים ב-ListCount **בלולאה כדי לבדוק תוכן מספר לא ידוע של פריטים, חשוב לזכור שהמערך** List **מתחיל באפס, ולכן הקוד צריך להיראות כדלהלן:**

```
For i =0 To List1.ListCount - 1
	'Process List1.List(i)
Next i
```

הערה:

אם שום פריט לא נבחר, ערך ListIndex **הוא 1-.**

להדגמת מאפיין ListIndex בפעולה, עצור את תוכנית הדוגמה והוסף את הקוד הבא לאירוע Click של תיבת הרשימה:

```
Msgbox List.List(List1.ListIndex)
```

הרץ שוב את התוכנית, וכשאתה לוחץ על פריט, הוא מוצג בתיבת ההודעה.

דרך אחרת לשליפת ערכו של פריט נבחר על ידי המשתמש מתיבת רשימה, היא בדיקת המאפיין Text של התיבה. מאפיין זה מכיל את הפריט מכל שורה ממוקדת בתיבת הרשימה. במקרה של תיבת רשימה פשוטה מחזיר המאפיין Text את השורה שעליה לחץ המשתמש. במקרה ששום שורה לא נבחרה, מחזיר המאפיין מחרוזת ריקה ("").

להסרת פריט מתיבת הרשימה השתמש בשיטת RemoveItem, בציון אינדקס המערך של הפריט המוסר. להדגמה הוסף את שורת הקוד הבאה לאירוע KeyDown בתיבת הרשימה:

```
Private Sub List1_KeyDown(KeyCode As Integer, Shift As Integer)

	If KeyCode = vbKeyDelete And List1.ListIndex <> -1 Then
		List1.RemoveItem List1.ListIndex
	End If
```

```
End Sub
```

הרץ את התוכנית עם השינויים החדשים ותוכל לסלק פריט מהרשימה על ידי בחירתו והקשה על המקש Delete.

שימוש במערך הרשימה

שיטת AddItem והשימוש בה להוספת פריט לרשימה כבר נדונו בפרק זה. ניתן גם להשתמש בפרמטר הרשות Index בצירוף עם שיטת AddItem. הפרמטר מציין את המקום במערך הרשימה, שבו אמור הפריט הנוסף להופיע. מציינים את ערך האינדקס של הפריט במערך, שלפניו מבקשים שהערך החדש יופיע. למשל, אם ברשימה חמישה פריטים, ורוצים שהפריט החדש יופיע לפני הפריט השלישי, משתמשים בקוד הבא:

```
list1.AddItem "Corvette", 2
```

מיון פריטים

בקטע הקודם הודגמה הוספה ומחיקה של פריטים מרשימה. בנוסף הודגמה הוספת פריט במיקום מוגדר על ידי קביעת האינדקס. אך מה עם מיון הפריטים לפי אלף-בית?

למזלנו זוהי משימה פשוטה. למיון הרשימה יש לקבוע את המאפיין Sorted כ-True. לאחר מכן יופיעו הפריטים ברשימה בסדר האלף-בית, בלי תלות בסדר הזרמתם. האינדקסים של הפריטים מותאמים עם הזנתם, כדי להבטיח הופעתם בסדר המבוקש.

הערה:

זכור, בסידור לפי האלף בית מופיע המספר 2 אחרי 11.

כושר מיון מובנה זה הוא מאפיין נאה של תיבת הרשימה, באפשרו מיון נתונים מהיר. אולם, דרוש לציין את המאפיין Sorted בעת התכנות. אם דרוש לאפשר למשתמשים להפעיל או להפסיק את המיון בעת הרצת התוכנית, ניתן להשתמש בשני פקדים של תיבת הרשימה, ולהשתמש במאפיין Visible, כך שרק אחד מהם יוצג בו-זמנית.

הערה:

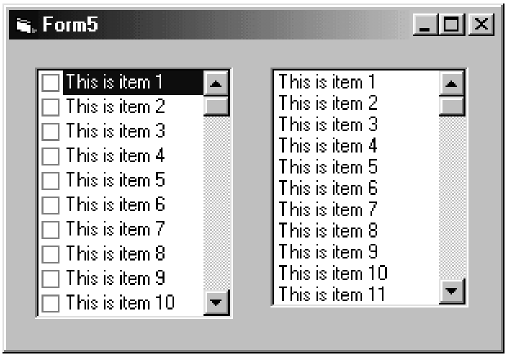
המאפיין Sorted לא ניתן לשינוי בזמן ריצת התוכנית, ניתן רק לראותו. כלומר הוא במצב ReadOnly בזמן ריצה.

קביעת ההופעה של תיבת רשימה

בנוסף לתיבת הרשימה ה"פשוטה" (תרשים 4.8), קיימים סגנונות הצגה נוספים. ניתן לגרום לתיבה להיראות כסדרת תיבות סימון, או להכליל בה כמה טורי פריטים.

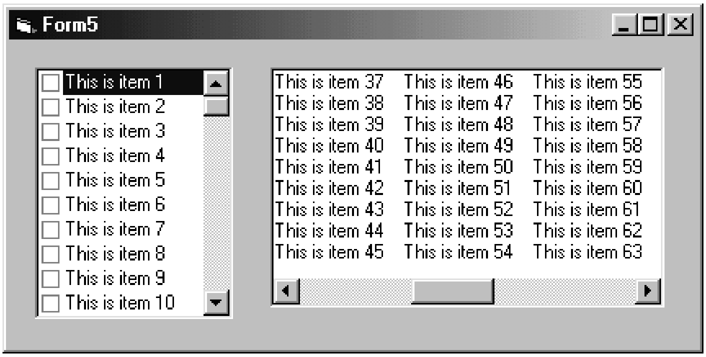
בחירת סוג הרשימה מתאפשרת על ידי קביעת המאפיין Style. שני המצבים של המאפיין הם Standard - 0, Checkbox - 1, כמודגם בתרשים 4.9. המאפיין ניתן לשינוי

גם תוך כדי ריצת התוכנית, על ידי קביעת ערכו כאחד משני הקבועים הפנימיים vbListBoxStandard או vbListBoxCheckBox בהתאמה.



תרשים 4.9: תיבות רשימה בסגנון תיבות סימון מספקות למשתמש דרך אינטואיטיבית לבחירת יותר מפריט אחד

דרך נוספת לשינוי מראה הרשימה היא בשימוש במאפיין Columns. ערך המחדל של המאפיין הוא 0. ערך זה מציג את תיבת הרשימה התקנית שנדונה לפני כן. קביעת הערך ל-1 גורמת להצגת הרשימה בצורת עמודה אחת בכל זמן נתון, אך עם גלילת טורים אופקית, לא אנכית. קביעת הערך למספר גדול מ-1 גורמת להצגת הרשימה בו-זמנית של מספר עמודות השווה לערך האינדקס (לדוגמה, הערך 2 יגרום להצגת שתי עמודות). כשהרשימה מוצגת במספר עמודות, הגלילה היא אופקית. ניתן להשתמש במאפיין Columns גם בעיצוב הרשימה התקני וגם בעיצוב בצורת תיבת הסימון. תרשים 4.10 מציג רשימה חד ורב-טורית. הרשימות פועלות באותה דרך, בלי תלות במספר העמודות.



תרשים 4.10: ניתן גם ליצור רשימות עם יותר מעמודה אחת

עבודה עם בחירות מרובות

לעיתים יש לאפשר למשתמשים לבחור יותר מפריט אחד מתוך רשימה. תיבת הרשימה תומכת באפשרות כזאת עם המאפיין MultiSelect. למאפיין זה שלושה מצבים אפשריים: None - 0, Simple - 1, וכן extended - 2.

- מצב המחדל None - 0, פירושו שבחירה מרובה אינה מותרת, ותיבת הרשימה יכולה לקבל רק ערך אחד בו-זמנית. שני המצבים האחרים מאפשרים בחירה מרובה. ההבדל ביניהם הוא בדרך שהם מאפשרים למשתמש לבצע את הבחירה.
- במצב Simple - 1 יכול המשתמש ללחוץ על פריט בעזרת העכבר כדי לבחור בו, או ללחוץ על פריט נבחר כדי לבטל את הבחירה. בשימוש במקלדת הוא יכול להשתמש במקשים כדי להזיז את הסמן להזזת המיקוד (הגבול המקווקו) לפריט מסוים, ואז להקיש על מקש הרווח לביצוע הבחירה או לביטולה.
- המצב השני של MultiSetting, מצב Extended - 2, מורכב יותר. במצב זה יכול המשתמש להשתמש בטכניקות התקניות של Windows לבחירה מהירה של פריטים מרובים. ניתן לבחור טווח פריטים בלחיצה על הפריט הראשון בטווח ואז תוך החזקת מקש Shift ללחוץ על הפריט האחרון בטווח. כל הפריטים בין הראשון לאחרון נבחרים. כדי להוסיף או להוריד פריט בודד מהרשימה, על המשתמש ללחוץ על הפריט, תוך החזקת מקש Ctrl.

קבלת הבחירה מתיבת רשימה מרובת אפשרויות שונה במקצת מאשר בבחירת פריט אחד. מכיון שמאפיין ListIndex פועל רק על בחירת פריט בודד, לא ניתן להשתמש בו. במקום זאת, יש לבחון כל פריט ברשימה, לבדיקה אם נבחר.

האינדיקציה אם פריט נבחר ניתנת במאפיין תיבת הרשימה Selected. כמו המאפיין List שהוזכר קודם, Selected הוא מערך, שיש בו אלמנט לכל פריט ברשימה. ערך המאפיין Selected לכל פריט הוא True (הפריט נבחר), או False (הפריט לא נבחר).

בנוסף יש צורך לדעת כמה פריטים ברשימה, כך שיהיה אפשר לכוון את הלולאה לבדיקת כל אפשרויות הבחירה. המידע הזה נכלל במאפיין ListCount. הקוד הבא מדפיס בטופס שם של כל פריט נבחר מהרשימה:

```
numitem = Fruits.ListCount
    For I = 0 to numitem - 1
        If Fruits.Selected(I) then Form1.Print Fruits.List(I)
    Next I
```

שמירת הנתונים האחרים ברשימה

נניח שמבקשים להציג למשתמשים רשימה של שמות הפריטים שבה יש משמעות, כגון רשימת שמות, אך נדרש שהרשימה תזכור מספר הקשור לכל שם. המאפיין ItemData של תיבת הרשימה הוא בבסיסו מערך של מספרים שלמים ארוכים, מספר אחד לכל פריט שנכלל בתיבת הרשימה. ItemData זוכר את המספר הקשור לפריט המסוים, בלא קשר למיקומו בתיבה. זה קורה גם אם המאפיין Sorted של התיבה הוא True, שפירושו כי הפריטים אינם רשומים בהכרח בתיבה לפי סדר הזרמתם אליה. למשל, האלמנט של מערך ItemData הקשור בפריט הראשון בתיבת הרשימה List1, ניתן לגישה כ-List1.ItemData(0). הגישה לאלמנט המערך של הפריט התורן הנבחר היא דרך List1.ItemData(List1.ListIndex) (יש לזכור כי המאפיין ListIndex מדווח איזה פריט הוא הנבחר התורן).

כשלתיבת הרשימה נוספים פריטים, נוצר במערך ItemData אלמנט הקשור אליו. הזנת הערך המתאים למקום המתאים במערך ItemData היא כמובן תפקידו של מזין הפריטים. אם כך איך תדע התוכנית באיזה מיקום בתיבת הרשימה הושם הפריט? Visual Basic הופכת זאת למשימה פשוטה. מאפיין NewIndex מאחסן את מספר האינדקס של הפריט האחרון שנקלט ברשימה. הקוד הבא מוסיף לקוח חדש לתיבת רשימה ממוינת ואז מוסיף את מספר חשבון הלקוח לאלמנט המתאים של המערך הקשור בו ItemData :

```
    lstCustomers.AddItem "Thomas, June"
    lst.Customers.ItemData(lstCustomers.NewIndex) = 21472301
```

עכשיו יכולה התוכנית לאפשר למשתמש לבחור מתוך תיבת רשימה המכילה אלמנטים שיש להם מובן (שמות), אך העיבוד ברקע יכול להתבצע עם המספר הקשור אליו, מה שמקל על פעולת המחשב. גישה זו מודגמת בשגרת האירוע Click של תיבת הרשימה :

```
Private Sub lstCustomers_Click()
    Dim lgThisCust = lstCustomers.ItemData(lstCustomers.ListIndex)
    Call LookUpAccount(lgThisCust)
End Sub
```

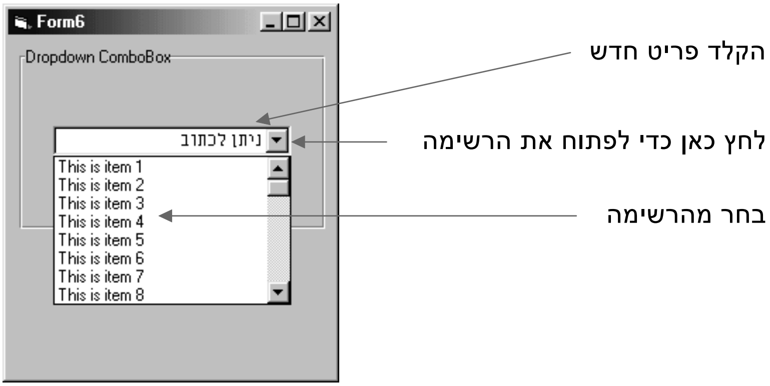
הערה:

LookUpAccount הינה פונקציה שמשתמשת במספר החשבון של הלקוח.

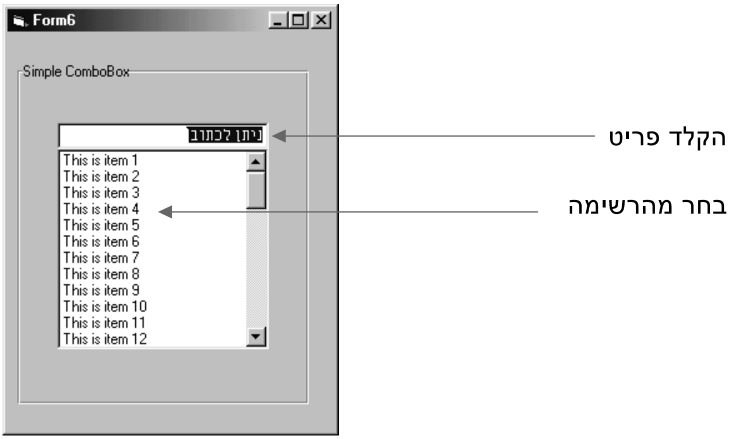
התיבה המשולבת

פקד נוסף המאפשר הצגת רשימות למשתמשים הוא **תיבה משולבת** (Combo Box).
השימוש בתיבה המשולבת אפשרי בשלוש צורות:

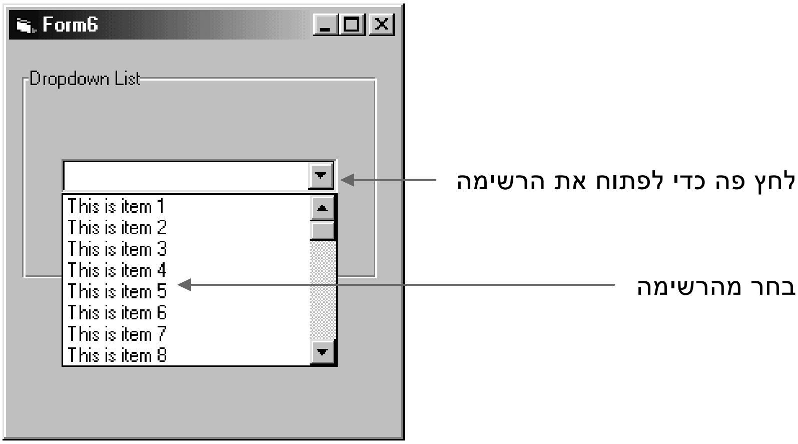
- **תיבה משולבת נפתחת** (Drop Down Combo Box)**.** מציגה למשתמש תיבת טקסט משולבת ברשימה נפתחת (ראה תרשים 4.11). המשתמש יכול לבחור פריט מהרשימה הנפתחת, או להקליד פריט חדש בתחום תיבת הטקסט.
- **תיבה משולבת פשוטה** (Simple Combo Box)**.** מציגה תיבת טקסט ורשימה שאינה נפתחת (ראה תרשים 4.12). כמו בתיבה המשולבת הנפתחת, יכול המשתמש לבחור פריט מהרשימה, או להוסיף פריט חדש בתיבת הטקסט.
- **רשימה נפתחת** (Drop Down List)**.** מציגה תיבת רשימה נפתחת. המשתמש יכול לבחור פריט מהרשימה הנפתחת, אך אינו יכול להוסיף פריט שאיננו ברשימה.



תרשים 4.11: בתיבה המשולבת הנפתחת ניתן לבחור פריט מהרשימה הנפתחת, או להוסיף פריט חליפי על ידי הקלדה בתיבת הטקסט



תרשים 4.12: התיבה המשולבת הפשוטה פועלת כמו התיבה המשולבת הנפתחת, פרט לכך שהרשימה מוצגת באופן קבוע



תרשים 4.13: ברשימה הנפתחת אין אפשרות להקליד פריט חליפי

הערה:

סוג חדש של תיבה משולבת, ImageCombo, נוסף בגירסה 6. פקד זה נדון בפרק 12 **הפקדים המשותפים של Microsoft**.

לתיבה המשולבת יש הרבה מן המשותף עם תיבת הרשימה. שתיהן משתמשות בשיטות AddItem, RemoveItem, Clear, לשינוי תוכן הרשימה. בשתיהן ניתן להציג רשימה ממוינת או בלתי ממוינת. שתיהן תומכות במערך ItemDate ובמאפיין NewIndex. ישנם אולם דברים מסוימים שתיבה אחת מסוגלת לבצע והשנייה לא.

התיבה המשולבת איננה תומכת בבחירת פריטים מרובים. יתרונה העיקרי של התיבה המשולבת הוא במתן אפשרות למשתמש להזין אפשרויות בחירה שלא נכללו מראש ברשימה. השיטה דומה לשימוש בפתק בחירות, שניתן לבחור בו במועמד מתוך רשימה, ואפשר גם להוסיף בו שם של מועמד נוסף, שלא נכלל בה.

הערה:

הרשימה הנפתחת איננה מאפשרת הוספת פריטים לרשימה על ידי המשתמש. יתרונה על תיבת רשימה פשוטה הוא בכך שהיא תופסת פחות מקום.

הקטעים הבאים מסבירים כיצד להשתמש בצורות השונות של התיבה המשולבת. הרשימה הנפתחת נדונה ראשונה, כי היא הפשוטה מכל סגנונות התיבה המשולבת.

יצירת רשימה נפתחת

רשימה נפתחת (Drop Down List) פועלת כמו תיבת רשימה. ההבדל העיקרי הוא, שרשימה נפתחת תופסת פחות מקום בטופס. כשמשתמש רוצה לבחור פריט מהרשימה, הוא לוחץ על החץ למטה הממוקם בימין התיבה, כדי לפתוח את הרשימה. לאחר הפתיחה ניתן לבחור פריט בלחיצה עליו. לאחר הבחירה נסגרת הרשימה כמו תריס נגלל של חלון, והפריט הנבחר מופיע בתיבה.

המפתח יוצר רשימה נפתחת על ידי ציור תיבה משולבת על הטופס וקביעת המאפיין Style לערך Dropdown List - 2. ניתן אז להוסיף פריטים לרשימה תוך שימוש במאפיין List, כפי שהדבר נעשה בתיבת רשימה. יש לזכור כי משתמשים אינם יכולים לבחור פריטים שאינם כלולים ברשימה כשעובדים עם סגנון זה.

קביעת הבחירה ההתחלתית

בתלות ביישום הנבחר, ייתכן שהמפתח ירצה לקבוע את הפריט ההתחלתי בתיבה המשולבת. ניקח לדוגמה תוכנית המבקשת לדעת את אזרחות המשתמש. התוכנית כוללת רשימת אפשרויות לבחירה, אך מעוניינים להציג כבחירה ראשונה את האפשרות "אזרח ארצות הברית", כי זו צפויה להיות בחירתם של רוב המשתמשים.

קביעת הערך ההתחלתי נעשית בשימוש במאפיין ListIndex. לדוגמה, אם "אזרח ארצות הברית" הוא הפריט הרביעי ברשימה, ניתן להשתמש בקוד הבא לגרום לו להיבחר מראש (זכור כי ListIndex מתחיל את הספירה מאפס):

```
Combo1.ListIndex = 3
```

משפט זה גורם לפריט הרביעי ברשימה להיות מוצג כשהתיבה המשולבת מופיעה לראשונה (זכור: האינדקסים של הרשימה מתחילים ב-0). ניתן לקבוע את הבחירה הראשונית בכל שלושת סוגי התיבות המשולבות. ניתן גם לקבוע את הבחירה הראשונית של התיבה המשולבת על ידי קביעת המאפיין Text לערך הרצוי. אם הבחירה ההתחלתית אינה נקבעת דרך קביעת האינדקס, מוצג הטקסט הנכלל במאפיין Text.



אזהרה:

הערך ההתחלתי של המאפיין Text הוא שם התיבה המשולבת. אם אינך רוצה ששם זה יופיע בהתחלה בתיבה המשולבת, עליך לקבוע את אחד המאפיינים ListIndex בקוד או למחוק את תכולת המאפיין text של הפקד.

עבודה עם אפשרויות בחירה שאינן ברשימה

הרשימה הנפתחת שימושית להצגת אפשרויות אחדות בשטח מצומצם. אולם, כוחה האמיתי של התיבה המשולבת הוא במתן אפשרות למשתמשים לבחור באפשרויות שאינן נכללות ברשימה. אפשרות זו קיימת גם בשני הסוגים השונים של התיבות המשולבות - הפשוטה והנפתחת. שני הסגנונות מספקים רשימת פריטים ממנה ניתן לבחור, ושתיהן מאפשרות הזנת ערכים אחרים. ההבדל בין שני הסגנונות הוא בדרך הגישה לפריטים הנכללים כבר ברשימה.

ניתן ליצור **תיבה משולבת פשוטה** (Simple Combo Box) על ידי הצבת הפקד על הטופס וקביעת המאפיין Style לערך Simple Combo - 1. בתיבות משולבות פשוטות יכול המשתמש לגשת לפריטי הרשימה בעזרת העכבר או מקשי החיצים. אם המשתמש איננו מוצא את מבוקשו ברשימה, הוא יכול להזין אפשרויות נוספות.

תיבה משולבת נפתחת (Drop Down Combo Box) פועלת כשילוב של רשימה נפתחת ותיבה משולבת פשוטה. בחירת פריט מהרשימה נעשית כמו ברשימה הנפתחת, אך ניתן להוסיף ערכים שאינם ברשימה. יוצרים תיבה משולבת נפתחת על ידי קביעת ערך המאפיין Style לערך Dropdown Combo - 0. התיבה המשולבת הנפתחת היא ברירת המחדל של המאפיין Style.

באשר לתיבה המשולבת הפשוטה ולתיבה המשולבת הנפתחת, ניתן להשתמש בהן במאפיין List להוספת פריטים לרשימת האפשרויות לבחירה. ניתן גם, כשהתוכנית רצה, לשנות את הרשימה בכל הסגנונות של התיבה המשולבת. כמו לגבי פקד תיבת הרשימה, השינוי נעשה בשימוש בשיטות AddItem, RemoveItem, Clear.

פקדים למטרות מיוחדות

הקבוצה האחרונה של הפקדים הפנימיים נקראת **פקדים למטרות מיוחדות**, מכיון שאין הם מתאימים להשתלבות בשום קבוצת פקדים אחרת. בהמשך תגלה בוודאי שיש להם שימוש רב.

להלן סוגי הפקדים שיידונו בקטע זה:

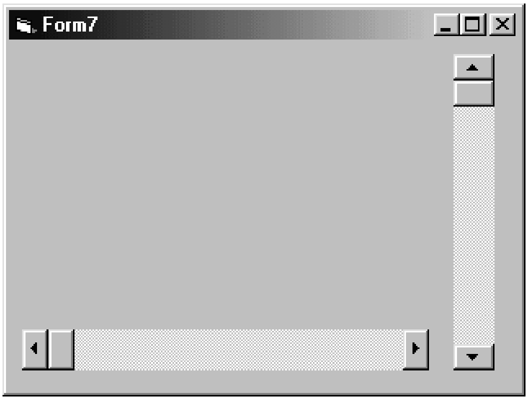
- פקדי **פס גלילה** (ScrollBar).
- פקדי **שעון עצר** (Timer).
- פקדי **מסגרת** (Frame).

פסי גלילה

מי שהשתמש במערכת ההפעלה Windows כבר מכיר את פסי הגלילה. פסי הגלילה מספקים אמצעי גרפי לבקרת המיקום בתוך מסמך, או לכיוון עוצמת הקול של כרטיס הקול במחשב.

פסי גלילה פועלים כמו וסתי עוצמת הקול במערכות סטריאו בעולם האמיתי, הניתנים לכיוון לכל נקודה בין ערכי מקסימום ומינימום מסוימים. מנקודת מבטו של המפתח פקדי פס הגלילה של Visual Basic מחזירים ערך מספרי. המתכנן אחראי להשתמש כראוי בערכים אלה ביישום.

Visual Basic מספקת שני סוגים של פס גלילה להזנת ערכים נומריים: **פס גלילה אופקי** (Horizontal) **ופס גלילה אנכי** (Vertical). שני פקדים אלה מוצגים בתרשים 4.14. ההתייחסות לפס הגלילה האופקי והאנכי היא HScrollBar ו-VScrollBar בהתאמה.



תרשים 4.14: הפקדים VScrollBar ו-HScrollBar יכולים לשמש להזנת נתונים והצגתם בדרך גרפית

ההבדל היחידי בין שני הפקדים הוא בכיוון הפס על הטופס. הקטע הבא דן בפקד HScrollBar, אך הוא מתאים גם לפקד VScrollBar.

קביעת פס הגלילה

כדי להשתמש בפס גלילה, משתמשים במאפיינים לקביעת הטווח ואחזרת הערך. שלושת המאפיינים החשובים, הזמינים לשימוש בעת התכנות ובעת ההרצה, הם:

- Value - שולף או קובע את הערך המספרי הנוכחי.
- Min - שולט על ערך המינימום של המאפיין Value.
- Max - שולט על ערך המקסימום של המאפיין Value.

כדי להשתמש במאפיינים אלה, יש ראשית לקבוע את המאפיינים Min ו-Max לערכים המייצגים את הטווח המבוקש, ואז לגשת למאפיין Value, למציאת ערכו. טווח אופייני הוא מ-0 עד 100, כשהמשתמשים מכניסים מספרים כאחוזים.

הערה:

פס הגלילה יכול לקבל מספרים שלמים בתחום שבין ‎-32,768‎ ל-32,767. אולם, הערכים התקפים של המאפיין Value תלויים בערכי המאפיינים Min ו-Max. יש לזכור זאת בעת קביעת ערכי פס גלילה מתוך קוד, מכיון שערך מחוץ לטווח גורם לשגיאה.

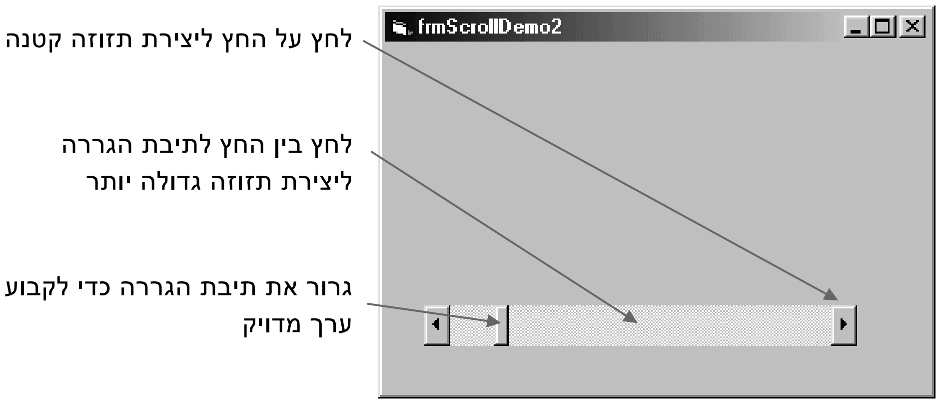
האירוע Change של פס גלילה מופעל כשהמאפיין Value משתנה. כדי לנסות פס גלילה פתח בפרויקט EXE תקני. מקם פס גלילה ותיבת טקסט על הטופס. אחר כך הזן את הקוד הבא באירוע Form_Load ובאירוע HScrollBar_Change.

```
Private Sub Form_Load()
	HScroll1.Min = Me.Left
	HScroll1.Max = Me.ScaleWidth
End Sub
Private Sub HScroll1_Change()
	Text1.Left = HScroll1.Value
End Sub
```

הרץ את התוכנית וקבע לפס הגלילה ערכים שונים על ידי גרירה, לחיצה ושימוש במקשי החיצים, Home וכן End. מיקום תיבת הגלילה בטופס אמור להשתנות, בהתאם לערכים הנוכחיים של פס הגלילה.

בקרת גודל שינוי הערך

מי שהשתמש בפסי גלילה במעבד תמלילים או בתוכנית אחרת, יודע שלחיצה על אחד החיצים המופיעים בשני צידי הפס, יוצרת הזזה קטנה בכיוון החץ הלחוץ, ולחיצה בין החץ ותיבת הגררה גורמת להזזה גדולה יותר. פקדי פס הגלילה עובדים באותו אופן, וניתן לקבוע בכמה מספרים ישתנה המיקום בכל סוג לחיצה. המספרים שהוזנו בפס הגלילה נכללים במאפיין Value. מאפיין זה משתנה בכל לחיצה על פס הגלילה וגרירת תיבת הגררה, כמצוין בתרשים 4.15.



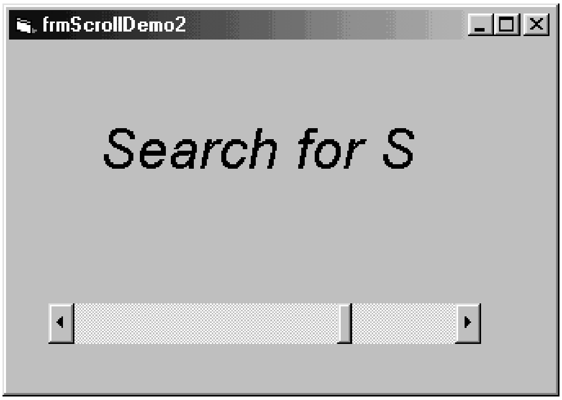
תרשים 4.15: לחיצה על חלקים שונים של פס הגלילה משנה את ערכו במידה שונה

המידה שבה גדל או קטן Value כאשר לוחצים על החץ, נשלטת על ידי המאפיין SmallChange, אלא אם כן חרג מהגבולות שנקבעו על ידי המאפיין Min או Max. SmallChange מאפשר שליטה עדינה ביותר על המספרים המוזנים. ערך המחדל שלו הוא 1, המתאים בדרך כלל לשימוש לרוב המטרות.

בלחיצה בין החץ לתיבת הגררה, יכול המאפיין Value להשתנות בכמות שונה מן השינוי בזמן הלחיצה על החץ. כמות השינוי במצב זה נקבעת על ידי המאפיין LargeChange. ערך המחדל של LargeChange הוא 1. הערך שבו משתמשים למעשה תלוי ביישום. למשל, הערך 10 הוא מספר טוב לעבודה באחוזים.

טיפ:

כלל טוב הוא לקבוע את המאפיין LargeChange בגבולות חמישה עד עשרה אחוזים מהטווח הכולל (לדוגמה, ערך 50 לטווח של 0 עד 1000).



תרשים 4.16: שיתוף בין תוויות או תיבות טקסט לבין פסי גלילה, מציג למשתמשים את הטווח ואת הערך הנוכחי

החלק הבעייתי הוא לדעת היכן במדויק למקם את שורות הקוד. כדי להראות את הערכים בכל רגע, דרוש למעשה למקם את המשפטים בשלושה אירועים:

- Form_Load - משתמשים באירוע זה להצגת הערך ההתחלתי של פס הגלילה, לאחר קביעת הטווח.
- Change - אירוע זה מופעל עם הלחיצה על לחצן העכבר בטווח שבין החיצים ותיבת הגררה. האירוע Change מופעל גם בכל פעם שלוחצים על אחד החיצים או על פס הגלילה.
- Scroll - אירוע זה פועל במשך גרירת תיבת הגררה של פס הגלילה. אירוע זה מאפשר להציג את הערך או לפעול כל עוד המאפיין Value עובר שינוי, אך לפני הפעלת האירוע Change.

כדי לנסות את דוגמת אלף-בית שמופיעה בתרשים 4.16, צור פרויקט EXE תקני. מקם פס גלילה ותוויות על הטופס. אחר כך הזן את הקוד המופיע בתוכנית 4.1.

תוכנית 4.1: SCROLLDEMO.VBP - השימוש בפס גלילה

```
Private Sub Form_Load()
	' Set Max, then Min properties to numeric values
	HScroll1.Max = Asc("Z")
	HScroll1.Min = Asc("A")

	'Display the initial value
	Label1.Caption = Chr$(HScroll1.Value)
End Sub
Private Sub Hscroll1_Change()
	Label1.Caption = "Search for " & Chr$(HScroll1.Value)
	' Insert code here to actually perform the db search
End Sub
Private Sub HScroll1_Scroll()
	Label1.Caption = "Release to select " & Chr$(HScroll1.Value)
End Sub
```

הרץ את התוכנית. שים לב לשינוי הטקסט בתווית בזמן שאתה מתפעל את פס הגלילה. בהכנסת קוד לאירועים המתאימים, ניתן, כמו בדוגמה שבתוכנית 4.1, להציג בכל רגע את ערכו המדויק של פס הגלילה, אך לנקוט בפעולה רק כאשר יידרש.

פקד שעון עצר

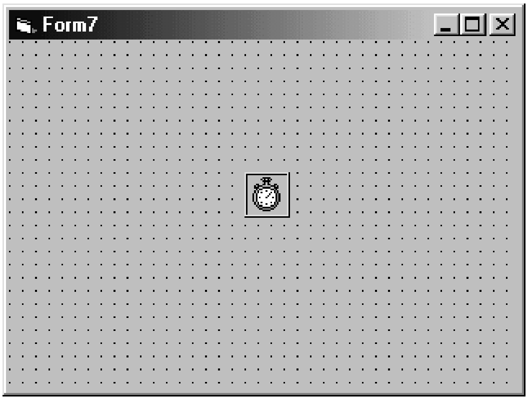
ממראה הסמל ניתן לנחש, שפקד שעון העצר פועל כשעון עצר או כשעון מעורר. קיימים שלושה הבדלים עיקריים בין קוצב זמן בעולם האמיתי לזה של Visual Basic :

- במקום להשמיע קול או ביפ, מבצע הפקד Timer קוד (המופיע באירוע Timer שלו) כשהושלם מרווח הזמן.
- הפקד Timer מבצע ספירה לאחור שוב ושוב, כל הזמן שהמאפיין Enabled הוא במצב True.
- הפקד Timer מתוכנן לעבודה עם פרקי זמן קצרים מאוד. קביעת הזמן הארוכה ביותר האפשרית בו היא מעט יותר מדקה.

כוונון שעון העצר

כדי לקבוע פקד שעון עצר, דרוש ראשית למקם אותו על הטופס, כמודגם בתרשים 4.17. פקד שעון העצר מופיע תמיד כסמל בשלב התכנון, אך איננו מוצג כלל בזמן ריצת התוכנית. כדי שהפקד יפעל, יש ראשית צורך למקם קוד באירוע Timer, ואז לקבוע את המאפיינים הבאים, בזמן התכנות או הריצה :

- קביעת המאפיין Interval.
- קביעת המאפיין Enabled כ-True.



תרשים 4.17: פקד שעון העצר מופיע כסמל, ללא תלות באיזה גודל שורטט

המאפיין Enabled פועל כמפסק המפעיל או מפסיק את שעון העצר. אם פקד שעון העצר מופעל, הקוד המצוין באירוע Timer מופעל בתום מרווח הזמן המצוין במאפיין Interval. ניתן לקבוע למאפיין Interval כל ערך שבין אפס ל-65,535. קביעת Interval כאפס מנטרלת את פקד שעון העצר. לגבי כל ערך אחר בטווח, המאפיין Timer מציין את הזמן באלפיות שנייה בין ההפעלות של האירוע Timer. במילים אחרות: אם רוצים שתעבורנה עשר שניות, יש לקבוע את הערך ל-10,000.

הערה:

אם המאפיין Enabled נקבע True בעת התכנות, שעון העצר מתחיל בספירה לאחור מייד עם טעינת הטופס המכיל אותו.

מכיון שהערך המקסימלי של Interval שווה בערך דקה, כיצד ניתן לקבוע פרקי זמן ארוכים יותר? ניתן לקבוע באירוע Timer קוד המשתמש בזמן המערכת כדי לקבוע אם פרק הזמן כבר חלף, כמודגם בתוכנית 4.2. הקוד בתוכנית גורם להדפסת הזמן בחלון אחת לחמש דקות.

תוכנית 4.2: TIMEREX.VBP - השימוש בפקד שעון העצר למרווחי זמן ארוכים

```
Option Explicit
Dim dtNextTime As Date

Private Sub Form_Load()
	' Set up the timer
	Timer1.Interval = 500
	Timer1.Enabled = True

	' We want to do something every 5 minutes
	dtNextTime = DateAdd("n", 5, Now)

End Sub
```

```
Private Sub Timer1_Timer()

       label1.caption = "Next Event:" & dtNextTime
       label2.caption = "Current Time:" & Now
              If Now >= dtNextTime Then
                     Timer1.Enabled = False

                     ' Do what you want here then update dtNextTime
                     Debug.Print "Timer event at " & Now
                     dtNextTime = DateAdd("n", 5, Now)

                     Timer1.Enabled = True
              End If
End Sub
```

הקוד בתוכנית 4.2 איננו משתמש בפקד Timer לביצוע אירוע. הוא משתמש בו כדי לבדוק את התאריך והשעה במערכת, לראות אם עליו לבצע אירוע. המפתח לתוכנית זו הוא הפונקציה DateAdd, המשמשת לחישוב השעה הצפויה בעוד חמש דקות.

הערה:

חלופה לפקד Timer, המבוססת על קוד בלבד, היא הפונקציה Timer, המחזירה את מספר השניות שעברו מחצות הלילה. ניתן לחסר את ערך הפונקציה המוחזר הנוכחי מהערך המאוחסן, כדי למצוא את מספר השניות שעברו, כמו בדוגמה הבאה:

```
Dim sngStart As Single
sngStart = Timer
' (perform some action)
Debug.Print "Elapsed time was " & CInt(Timer - sngStart) & " seconds."
```

יצירת אנימציה פשוטה

להדגמת קלות השימוש בפקד Timer, להלן הוראות ליצירת אנימציה פשוטה:

1. פתח פרויקט EXE תקני חדש ב- Visual Basic.
2. מקם פקד Timer ופקד תמונה על הטופס.
3. מחלון **Properties** של פקד התמונה, בשורת המאפיין **Picture**, בחר בלחצן הבנייה המזוהה לפי שלוש הנקודות (Ellipsis) שעליו.
4. כשמופיעה תיבת הדו-שיח Load Picture, בחר קובץ תמונה מהתיקיה Visual Basic\Graphics\Icon. לדוגמה בחר את הקובץ FACE03.ICO מהתיקיה Misc. פקד התמונה אמור להכיל עכשיו תמונה כלשהי.

5. הזן את הקוד הבא באירוע Timer, באירוע Form Load ובקטע ההצהרות:

```
Dim xChange As Integer
Dim yChange As Integer

Private Sub Form_Load()
       xChange = 100
       yChange = 100

End Sub

Private Sub Timer1_Timer()
       Image.Left = Image1.Left + xChange
       Image1.Top = Image1.Top + yChange

       If Image1.Left > Me.ScaleWidth Then xChange = xChange * -1
       If Image1.Left < 0 Then xChange = xChange * -1
       If Image1.Top > Me.ScaleHeight Then yChange = yChange * -1
       If Image1.Top < 0 Then yChange = yChange * -1
End Sub
```

6. פתח את חלון Properties של פקד Timer.

7. קבע למאפיין Interval את הערך 200 ולמאפיין Enabled את הערך True.

כשתריץ את תוכנית, אמורה התמונה לנוע על פני הטופס, ולשנות כיוון בכל פעם שהיא "נתקלת" בגבולותיו.

מסגרות

הפקד Frame הוא מיוחד, מכיון שהוא פקד מסוג הנקרא **מכולה** (Container). מכולות הן פקדים שבהם ניתן למקם פקדים אחרים, בדומה למסגרת תמונה אמיתית הסוגרת על (מכילה) תמונת האמן. כשמסתירים (בשימוש במאפיין Visible), או משנים מיקום פקד מכולה בטופס, הפקדים שהוא מכיל מושפעים גם הם. תכונה זו שימושית ביצירת "דפים" רבים של פקדים על טופס יחיד.

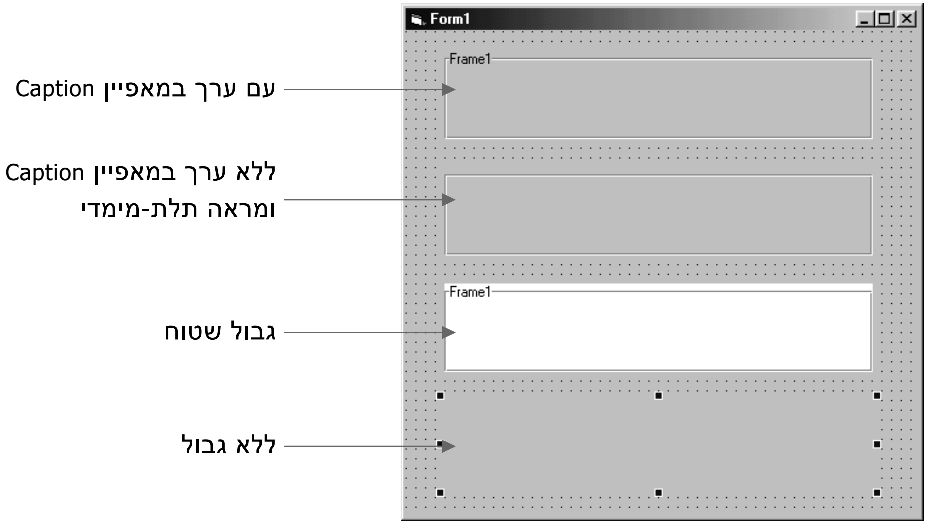
הערה:

פקדי מכולה יוצרים קשר חזותי שונה בין הטופס והפקדים הנכללים בו. אירועי הקוד וטווח המשתנים בטופס אינם מושפעים.

כמו לגבי כל הפקדים האחרים, הצעד הראשון בקביעת פקד מסגרת, הוא לצייר אותו בגודל המבוקש. אחר יש לקבוע למאפיין Name שם ייחודי. בשלב זה ניתן לקבוע מאפיינים אחדים השולטים על הופעת המסגרת. השפעתם מודגמת בתרשים 4.18:

- Caption - כשמוזן ערך למאפיין זה, הטקסט המוזן מופיע בפינת המסגרת הימנית העליונה. כיתוב זה יכול לשמש לזיהוי תוכן המסגרת, או לספק מידע תיאורי. אם מעוניינים במסגרת רציפה ללא כיתוב, יש למחוק את הכיתוב במאפיין Caption.

- Appearance - מאפיין זה קובע אם הגבול מוצג כקו יחיד בצבע אחיד, מה שנותן לפקד הופעה שטוחה, או שמשתמשים בקווים ליצירת אפקט תלת-מימדי למסגרת.
- BorderStyle - מאפיין זה קובע אם יוצג גבול מסביב למסגרת. אם המאפיין BorderStyle נקבע כ-None (משתמשים בערך 0), שום גבול אינו מוצג. גם הכיתוב אינו מוצג, כי גם הוא מהווה חלק מהגבול.



תרשים 4.18: השליטה בהופעת המסגרת מושגת בעזרת המאפיינים Appearance, BorderStyle ו-Caption

טיפ:

אם אין רוצים שהמסגרת תיגע בטקסט הכיתוב, מוסיפים רווח לפניו ואחריו.

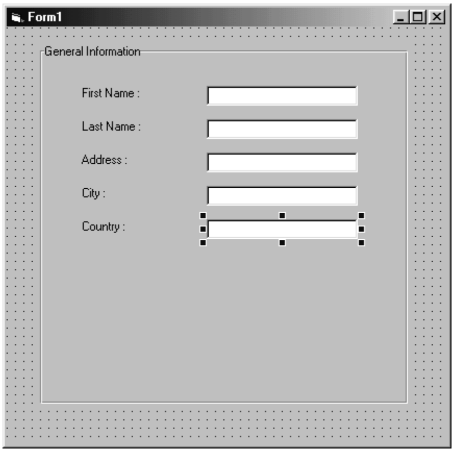
טיפ:

אם רוצים שהמסגרת תופיע ללא קו גבול, משאירים, בשלב התכנות, את הגבול על הטופס, ובשלב טעינת הטופס קובעים למאפיין BorderStyle את הערך 0. הצגת הגבול בשלב התכנות, מקלה על שמירת גבולות המסגרת.

עבודה עם פקדים במסגרת

לאחר ציור הפקד Frame, אפשר להתחיל למקם בו פקדים אחרים. ניתן למקם במסגרת (או בכל פקד מכולה אחר) כל פקד שמעוניינים בו. ניתן גם להציב מכולה בתוך מכולה. להצבת פקדים במסגרת, פשוט מציירים אותם ישירות בתוכה, בדומה לציור פקדים בטופס. (אם מוסיפים בלחיצה כפולה כלי מתיבת הכלים, חשוב שהמסגרת תיבחר לפני הוספת הפקד). לאחר מכן מציירים פקדים בתוך המסגרת ממש

כפי שהדבר נעשה בטופס. חשוב לוודא שבתחילת הציור ממוקם הסמן בתוך המסגרת, אחרת לא ייכלל בה הפקד. תרשים 4.19 מראה פקדי טופס כוח אדם אחדים מצוירים במסגרת.



תרשים 4.19: לפני שמתחילים לשרטט פקד, יש לוודא שהסמן ממוקם בתוך המסגרת

להלן שתי נקודות המתייחסות לפקדים ומסגרות:

- ראשית, אם על טופס מופיעים כבר פקדים, הם אינם נכללים במסגרת גם אם המסגרת מצוירת עליהם.
- שנית, כשמזיזים פקד במסגרת, לא ניתן להעביר פקד בגרירה ושחרור אל תוך מסגרת או החוצה ממנה.

ניתן לגרור פקד ולשים אותו על המסגרת כך שייראה שהפקד כלול במסגרת, אך זה איננו המצב. הדרך היחידה להעביר פקד מחלקי טופס אחרים אל תוך מסגרת היא בגזירה ובהדבקה. גזור את הפקד מהטופס, בחר במסגרת, ואז הדבק את הפקד במסגרת. (ניתן גם להעתיק פקד מחלק אחר של הטופס, ואז להדביקו במסגרת). תחילה מוצב הפקד בפינת המסגרת השמאלית העליונה, אך לאחר מכן ניתן לגרור אותו ולשחררו במקום אחר בתוך המסגרת. ניתן להשתמש בטכניקה זו גם להזזת פקדים מחוץ למסגרת. ברור שניתן להזיז כמה פקדים באותו זמן.



טיפ:

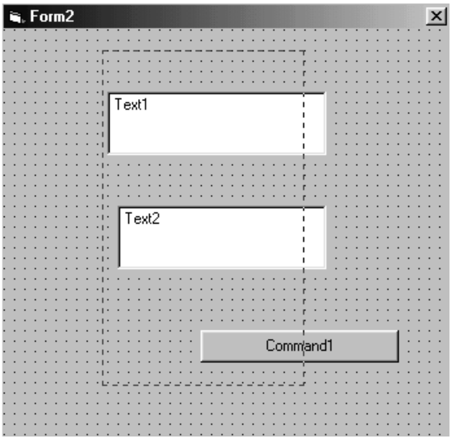
אם רוצים לוודא שהפקד כלול במסגרת, יש להזיז את המסגרת. אם הפקד זז עם המסגרת, הוא כלול בתוכה. אם אינו זז, ניתן לגזור ולהדביק אותו בתוך המסגרת.

עבודה עם מספר פקדים בשלב העיצוב

עד עכשיו ראינו כיצד ניתן להוסיף פקדים לטפסים ואיך לקבוע את מאפייני פקד אחד ברגע נתון. אולם, לעיתים יש צורך לעבוד עם מספר פקדים בו-זמנית. למשל, נניח שקיימת קבוצה של פקדי Label, והחלטת לשנות את הגופן שלהם, אך אינך רוצה לבחור ולשנות כל פקד תווית לחוד. למרבה המזל ניתן לפעול על כל הפקדים כקבוצה.

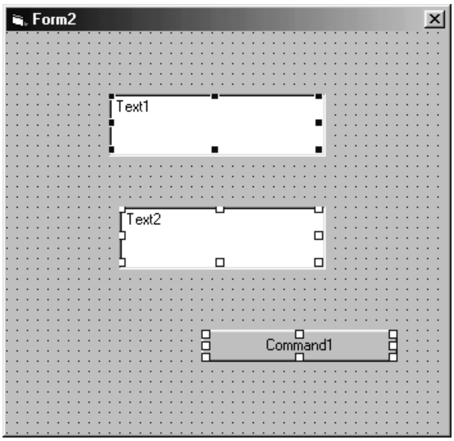
בחירת פקדים מרובים

הצעד הראשון בעבודה עם מספר פקדים הוא בחירת כל הפקדים שרוצים להזיזם או לשנותם. ניתן לבחור קבוצת פקדים בלחיצת עכבר על הטופס ובגרירתו. מסגרת מקווקוות מופיעה על הטופס, כמודגם בתרשים 4.20. השתמש במסגרת זו כדי לכלול בה את כל הפקדים שאתה בוחר בהם.



תרשים 4.20: בשימוש בעכבר ניתן בקלות לבחור יותר מפקד אחד

כשמשוחרר לחצן העכבר, נבחרו כל הפקדים הנמצאים בתוך התיבה. פקד נבחר מאופיין בתוספת נקודות אחיזה לשינוי גודל בכל פינה ובמרכז כל צלע שלו, כפי שניתן לראות בתרשים 4.21.



תרשים 4.21: נקודות אחיזה לשינוי גודל מציינות בחירת פקד אחד או יותר

הערה:

ניתן לבחור מספר פקדים גם על ידי החזקת מקש Ctrl תוך כדי לחיצה על כל אחד מהם. אם דרוש לבחור קבוצת פקדים הנכללים בתוך מסגרת או תיבת תמונה, חייבים לפעול בשיטת Ctrl ולחיצה, כי טכניקת התיבה המקווקוות אינה פועלת במקרים אלה.

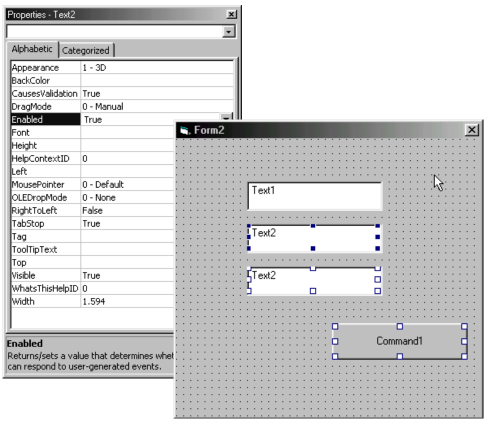
ניתן להוסיף או להוריד פקדים מקבוצה בלחיצה על הפקד תוך החזקת המקש Shift או Ctrl. ניתן להפעיל שיטה זו גם אם הבחירה הראשונית נעשתה בגרירה על ידי עכבר, כך שניתן לעדן את הבחירה ולהגיע בדיוק לאותם פקדים איתם מתכוונים לעבוד.

לאחר בחירת קבוצת הפקדים, ניתן להניע את הקבוצה כיחידה אחת, בלחיצה על אחד הפקדים מהקבוצה, וגרירת הקבוצה כולה. הפקדים שומרים תוך כדי ההזזה על מיקומם היחסי בתוך הקבוצה. ניתן גם לבצע פעולות עריכה, כמו מחיקה, גזירה, העתקה והדבקה, על הקבוצה כגוף אחד.

השימוש בחלון המאפיינים

בנוסף לאפשרות להזיז, לגזור, להעתיק, להדביק ולמחוק קבוצת פקדים, ניתן גם לעבוד עם מאפייניהם כקבוצה. למשל, אם רוצים לשנות את הגופן של קבוצת פקדי Label, בוחרים פשוט את קבוצת הפקדים, וניגשים למאפיין Font בחלון Properties. כשמשנים את המאפיין, מושפעים כל הפקדים שנבחרו. טכניקה זו שימושית כשרוצים לשנות מספר פקדים בבת אחת.

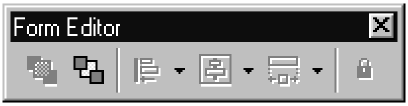
למרות שברור ששיטה זו עובדת עם קבוצת פקדים מאותו סוג (למשל קבוצת תוויות או קבוצת תיבות טקסט), נשאלת השאלה מה קורה כשהקבוצה כוללת פקדים מסוגים שונים. במקרה זה מציגה Visual Basic את המאפיינים המשותפים לכל הפקדים בקבוצה. בדרך כלל נכללים שם המאפיינים Top, Left, Height, Width, Font, ForeColor, Visible, Enabled. רק המאפיינים המשותפים לכל הפקדים ניתנים לעריכה. אולם, יכולת זו שימושית אם רוצים ליישר קבוצת פקדים לשמאל הטופס או בראשו. לשם כך יש פשוט לבחור את הקבוצה, ולקבוע את המאפיין Left (או Top) שלה. תרשים 4.22 מראה קבוצה נבחרת של פקדים שונים ואת החלון Properties המכיל את המאפיינים המשותפים.



תרשים 4.22: ניתן לשנות כקבוצה מאפיינים המשותפים לפקדים שונים

סרגל הכלים Form Editor

עריכת מאפיינים משותפים אינה הדרך היחידה לעבודה עם קבוצת פקדים. ניתן גם לעבוד עם סרגל הכלים **Form Editor** (ראה תרשים 4.23). ניגשים אליו על ידי בחירת View, Toolbars, Form Editor. טבלה 4.2 מציגה ומסבירה את לחצני סרגל הכלים:



תרשים 4.23: Form Editor הופך יישור פקדים וקביעת גודלם לפעולה פשוטה

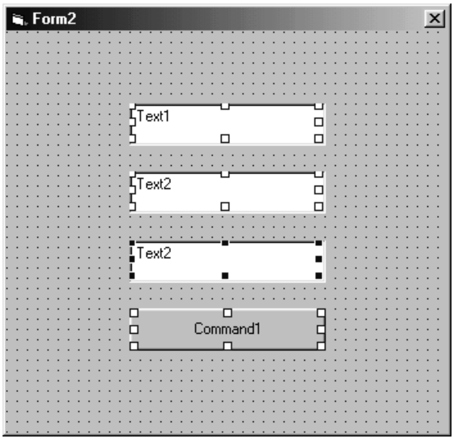
טבלה 4.2: לחצני סרגל הכלים Toolbar

שם	פעולה
Bring to Front	מעביר את הפקד הנבחר לתצוגה לפני הפקדים האחרים בטופס (יסתיר אותם)
Send to Back	מעביר את הפקד הנבחר אל מאחורי הפקדים האחרים בטופס (קרוב לוודאי שיוסתר על ידם)
Align	מיישר קבוצת פקדים
Center	ממרכז קבוצת פקדים
Make Same Size	משנה ומתאים גודל קבוצת פקדים
Lock Control Toggles	מונע הזזת פקדים או שינוי גודלם על ידי העכבר (ניתן עדיין לשנות את הפקדים דרך חלון Properties)

בשימוש בסרגל הכלים Form Editor ניתן לטפל במיקום ובגודל קבוצת פקדים. לחצן Align מאפשר יישור קבוצת פקדים לפי ימין, שמאל, בראש הטופס או בתחתיתו. למרות שיישור לשמאל או למעלה מתבצע ישירות על ידי קביעת המאפיין Left והמאפיין Top, אין אפשרות לבצע ישירות יישור לימין או לתחתית. הלחצן Align מאפשר גם מרכוז אנכי או אופקי של הפקדים. לא ניתן לבצע שינוי זה ישירות דרך המאפיינים. לבחירת סוג היישור המבוקש, לוחצים על לחצן החץ לימין הלחצן Align. פעולה זו גורמת להצגת תפריט, ממנו ניתן לבחור את היישור המבוקש. ניתן גם ליישר את הפקדים הנבחרים לפי הרשת. יישור לפי שיטה זו גורם לפינה השמאלית העליונה של כל פקד לזוז ולהיצמד לנקודת הרשת הקרובה אליו.

משימה אחרת שדרשה עבודה רבה בגרסאות מוקדמות של Visual Basic היא מרכוז פקדים על טופס. גם משימה זו הופכת לפשוטה בשימוש בסרגל הכלים Form Editor. ליד הלחצן Align מופיע לחצן Center. לחצן זה מאפשר למרכז את הפקדים אופקית או אנכית בטופס. הקבוצה ממורכזת כולה כאילו היתה פקד אחד בטופס, ומיקומו היחסי של כל פקד בקבוצה נשאר ללא שינוי. כמו ביישור, גם במרכוז בוחרים את האופציה הרצויה מתוך תפריט המשנה המופיע עם הלחיצה על החץ שליד הלחצן Center.

הלחצן Make Same Size מאפשר להשוות גובה ו/או רוחב של כל הפקדים בקבוצה. למרות שניתן לבצע זאת על ידי קביעת המאפיינים המתאימים, Form Editor מפשט את הפעולה. תרשים 4.24 מראה קבוצת פקדים המוגדלים באופן שווה וממורכזים אופקית.

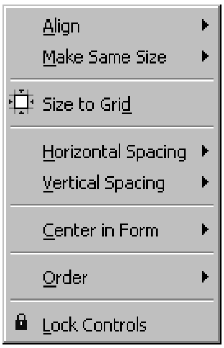


תרשים 4.24: סרגל הכלים Form Editor מקל על מרכוז פקדים ושינוי גודלם

טיפ:
אם לחצת על הלחצן הלא נכון בסרגל כלים Form Editor, ניתן לבטל את השינוי בלחיצה על הלחצן Undo בסרגל הכלים הסטנדרטי, או בהקשת Ctrl+Z.

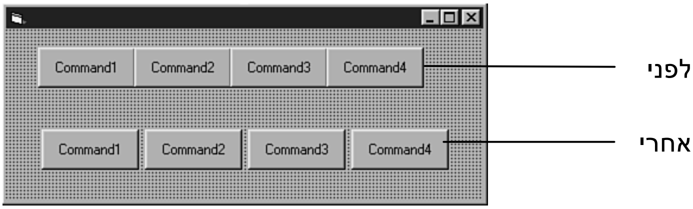
תפריט Format

תפריט **Format** של Visual Basic שימושי גם הוא בעבודה עם פקדים. תפריט זה מכיל, בנוסף לכל האפשרויות הכלולות בסרגל הכלים Form Editor, שלוש אפשרויות נוספות לעבודה עם פקדים מרובים: Horizontal Spacing, Vertical Spacing, Size to Grid. (ראה תרשים 4.25).



תרשים 4.25: תפריט Format מציע כלים לכוונון עדין של פקדים

האופציה Horizontal Spacing מאפשרת השוואת המרווחים בין הפקדים. השוואת מרווחים מעניקה מראה נקי לקבוצות פקדים כגון לחצני פקודה. אם הפקדים נראים קרובים מידי, ניתן לרווח אותם. המרווח גדל בנקודת רשת אחת. ניתן גם להקטין את המרווח או לסלקו לגמרי. ניתן לבצע את אותה פעולה גם בכיוון אנכי, על ידי בחירת Vertical Spacing. תרשים 4.26 מציג מראה "לפני" ו"אחרי" של קבוצת לחצני פקודה. המראה "אחרי" מציג אפקט של קביעת מרווחים שווים בין הלחצנים.



תרשים 4.26: מרווחים אופקיים שווים מעניקים הופעה מסודרת לקבוצת לחצנים

הפריט האחרון בתפריט Format הוא Size to Grid. בחירת אפשרות זו קובעת את המאפיינים Height ו-Grid של כל פקד נבחר, כך שיותאמו בדיוק למרווחי הרשת.

מספר פקדים במסגרת

כעת נלמד איך לבחור מספר פקדים מטופס, איך לפעול עם מאפייניהם המשותפים, או כיצד לטפל במשימות של יישור פקדים וקביעת גודלם. באותה גישה ניתן לטפל גם בפקדים בתוך מסגרת. ההבדל היחיד הוא בדרך שבה נבחרים הפקדים. כדי לבחור פקדים דרוש ראשית ללחוץ על הטופס כדי לבטל כל בחירה קודמת. לאחר מכן גוררים את העכבר מסביב לפקדים שבמסגרת, תוך החזקת מקש Ctrl. פעולה זו גורמת לבחירת קבוצת הפקדים. ניתן עדיין לבחור או לבטל בחירת פקדים אחרים על ידי לחיצה על הפקד תוך החזקת מקש Ctrl.

הערה:

כשבוחרים קבוצת פקדים, כולם חייבים להיות או בתוך מכולה או מחוץ לה. כמו כן, בחירת פקדים איננה יכולה לגשר בין מכולות.

לאחר בחירת מספר פקדים, ניתן לעבוד עם מאפייניהם המשותפים, או להשתמש בפקודות תפריט Format לשם קביעת מיקום הפקדים וגודלם.

אזהרה:

הפריט Center בפקודה Form של התפריט Format ממרכזת את הפקדים בטופס ולא במסגרת. התוצאה המתקבלת עלולה להיות שונה מן המבוקש.

עבודה עם אוסף הפקדים

עד עכשיו נדון הטיפול באוסף פקדים בסביבת תכנות. אך מה המצב בזמן הרצת התוכנית? כפי שכבר נאמר, ניתן לשנות לכל פקד את שם הפקד, שמות מאפיינים וערכם. אך האם חייבים לקבוע ערך כל פקד בקבוצה בנפרד? לא!

כל טופס כולל **אוסף** (Collection) הנקרא אוסף **פקדים** (Controls). אוסף זה מזהה כל פקד על הטופס. בשימוש באוסף זה ובצורה מיוחדת של לולאת For, ניתן לשנות מאפיין מסוים של כל פקד בטופס.

שינוי כל הפקדים

כדוגמה לשינוי פקדים נניח שרוצים לאפשר למשתמש לבחור את הגופן שבו ישתמשו כל הפקדים בטופס. הניסיון לקבוע כל טופס דרך קוד עלול להיות מסובך. אולם, בשימוש בלולאת For Each ניתן לשנות גופן של כל פקד בטופס בעזרת שלוש שורות קוד בלבד:

```
For each Control In Form1.Controls
	Control.Font = "Times New Roman"
Next Control
```



אזהרה:

קוד זה יפעל רק אם כל הפקדים בטופס תומכים במאפיין Font. לכמה פקדים, למשל הפקד Timer, לא קיים מאפיין כזה.

שינוי פקדים נבחרים

מה קורה כשרוצים לקבוע מאפיינים שאינם נתמכים על ידי כל פקדי הטופס? נניח למשל שרוצים לשנות צבע טקסט של הפקדים. ללחצן הפקודה אין מאפיין ForeColor, למרות שבסוגי פקדים אחרים הוא קיים. במקרים כאלה משתמשים משפט קוד אחר: משפט TypeOf. משפט זה מאפשר לקבוע מה סוגו של כל אובייקט. באופן כזה ניתן לבנות שיגרה המשנה את המאפיין ForeColor של כל הפקדים שאינם לחצני פקודה:

```
For Each Control In Form1.Controls
	If Not TypeOf Control Is CommandButton Then
		Control.ForeColor = &HFF
	End If
Next Control
```

מכאן...

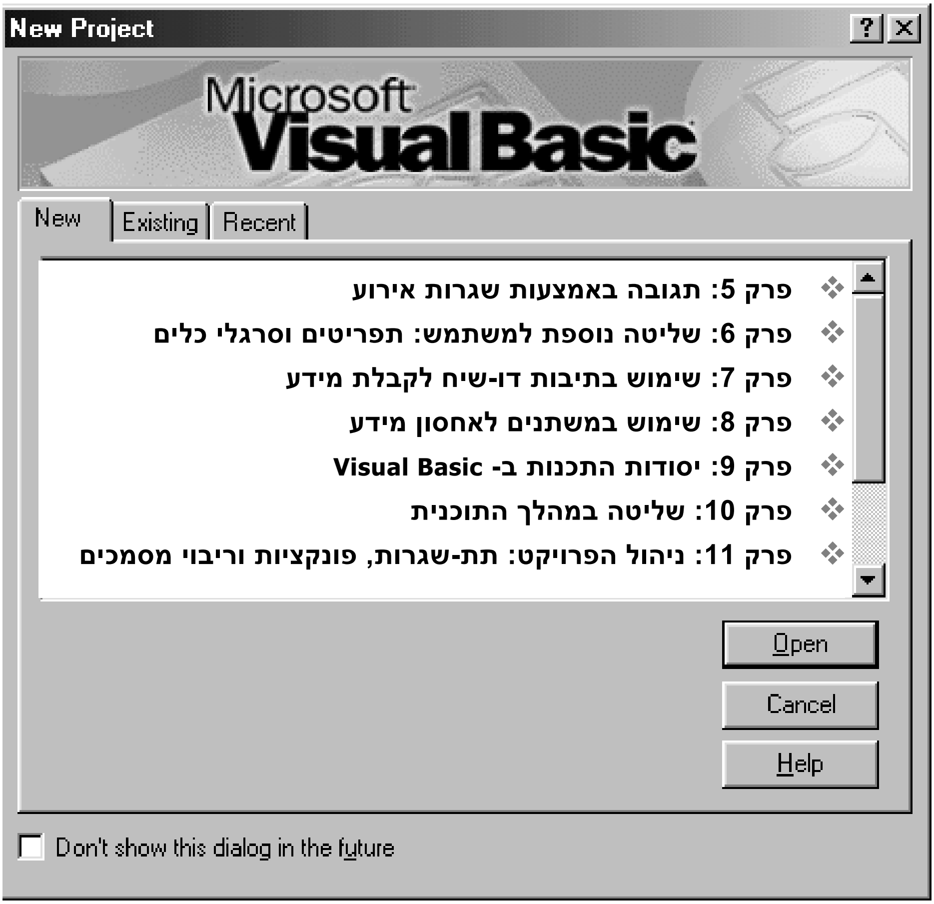
פרק זה הציג בפני הקורא את עולם הטפסים והפקדים. נלמד בו כיצד לשלוט בהופעת הטפסים והפקדים דרך המאפיינים שלהם, וכיצד להשתמש בשיטות לביצוע משימות. כמו כן הודגמו דרכי העבודה עם מספר פקדים בסביבת תכנות ובתוכניות.

מידע נוסף בנושאים שנדונו בפרק ניתן למצוא במקומות הבאים:

- ללימוד על פקדים נוספים, ראה פרק 6 **שליטה נוספת למשתמש: תפריטים וסרגלי כלים**, וכן פרק 12 **הפקדים המשותפים של Microsoft**.
- ללימוד על השימוש בפקדים במערך, ראה פרק 13 **עבודה עם מערכי פקדים**.
- כדי לגלות איך לסדר פקדים באופן נעים לעין, וכן ללימוד על כמה פקדים הקשורים לגרפיקה, ראה פרק 19 **שימוש במרכיבי התכנון הוויזואלי**.

חלק 2

תכנות ב- Visual Basic



5

תגובה באמצעות שגרות אירוע

מה בפרק?

- היכרות עם אירועים
- טיפול באירועים בתוכניות
- הבנת סדר האירועים

עם הופעת מערכות הפעלה גרפיות מונחות-אובייקטים כגון Windows ודומיה, השתנתה לחלוטין הדרך שבה המשתמשים תיקשרו עם תוכניות. עברו הימים אשר בהם רצו תוכניות טקסטואליות מתחילתן ועד סופן כמעט ללא התערבות המשתמש. בעזרת Windows, יש לנו המפתחים שליטה רחבה בפונקציות שאותן התוכנית אמורה למלא וכן הסדר שבו הפונקציות תתבצענה. אולם, למרות העובדה שתפיסה זו הפכה את התוכניות לקלות יותר למשתמש, צורת הכתיבה והפיתוח שלהן הפכה לקשה יותר. למפתח אין כיום שליטה מלאה על מהלך התוכנית. מפתחים בסביבה הפועלת על אירועים חייבים להכין את התוכנית למספר רב של פעולות אשר המשתמש אמור לבצע ובמקרים רבים יש צורך להגן על המשתמש מעצמו.

יישומים המבוססים על Windows הינם, מאובייקט היותם פעילים בסביבת Windows, מבוססי אירועים. פירושו של דבר הוא שמהלך ריצת התוכנית מבוסס על האירועים (Events) שקורים בזמן ביצועה. כמו שנראה, רוב האירועים הנזכרים הינם תוצאה ישירה של פעולות המבוצעות על ידי המשתמש. אולם, חלק מן האירועים יבוצעו על ידי אובייקטים אחרים כגון התוכנית עצמה.

הערה:

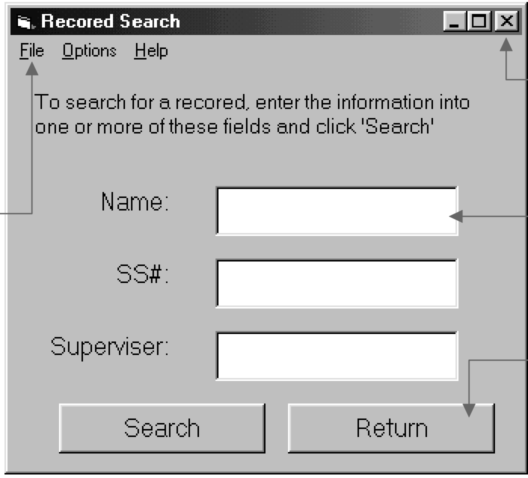
למרות שתוכניות מבוססות-אירועים היו קיימות עוד לפני הופעת Windows בשוק (תוכניות מקינטוש ואפילו כמה תוכניות DOS), הופעת Windows גרמה לסוג זה של תוכניות להיות שכיחות כיום.

אירועים - מבוא

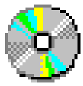
צורת החשיבה של תוכניות מבוססות-אירועים עלולה להיות חדשה לך, אולם הרעיון של תגובה לפי אירוע מסוים אמור להיות מוכר. רוב העולם יכול להיות מוגדר כפועל לפי אירועים. ניקח את הטלוויזיה כדוגמה. אנו יכולים לשנות את הערוץ מתי שנרצה, ובשימוש בשלטים הקיימים כיום ניתן גם לעבור לערוץ מסוים במקום לעבור דרך סידרה של ערוצים כדי להגיע לערוץ המבוקש. בנוסף, ניתן לשנות את עוצמת השמע של הטלוויזיה מתי שנרצה לעוצמה הרצויה לנו. נוכל גם להשתיק את השמע לגמרי על ידי לחיצה על לחצן ההשתקה. צורה זו של שליטה על איך ומתי דברים יתרחשו מהווה דוגמה טובה לצורת תכנות מונחה-אירועים. אנו (משתמשי הטלוויזיה) יוצרים אירוע (על ידי לחיצה על לחצן) אשר גורם לטלוויזיה לבצע פעולה המתאימה לאירוע שהתרחש. אנו שולטים גם בתזמון האירועים.

תוכניות המבוססות על Windows עובדות בצורה דומה. ב-Word, למשל, ניתן לשנות את סוג הכתב או הסגנון של פיסקה מסוימת. ניתן גם לסמן מקטע מסוים ולגרור אותו למקום אחר. כל אחת מפעולות אלו אפשרית תודות ליכולת התוכנה (Word) להגיב בהתאם לאירועים המתרחשים כתוצאה מפעולות המשתמש.

בזמן כתיבת תוכניות המבוססות על Windows, נרצה לבנות אותן בהתאם למשימות אותן אמורות התוכניות לבצע. לכן, אנו צריכים לספק למשתמשים לחצנים או תפריטים אשר יאפשרו להם לבצע את המשימות הדרושות. תרשים 5.1 מציג דוגמה לממשק המיועד לתוכנית המבוססת על אירועים.



תרשים 5.1: ממשק משתמש גרפי עובד יפה עם תכנות מונחה-אירועים



בתקליטור:

הדוגמה נמצאת בתקליטור ונקראת Record Search.

יתרון נוסף של תוכניות המבוססות על אירועים הוא שניתן להשתמש באירועים כדי לספק תגובה מיידית למשתמש. לדוגמה, ניתן לתכנת אירוע אשר יגרום לתוכנית לבדוק מידע אשר הוזן על ידי המשתמש ברגע שבו המשתמש סיים להזין אותו. במידה ויש בעיה עם המידע אשר הוזן על ידי המשתמש, הוא יקבל על כך התראה מיידית ותינתן לו האפשרות לתקן אותו. צורה זו של תגובה מטופלת באמצעות צורת תכנות כגון זו המופיעה בתוכנית 5.1 אשר מוודאת שהמשתמש הזין את המידע המבוקש.

תוכנית 5.1: Events.VBP - תוכנית הבודקת את שם המשפחה.

```
Private Sub txtLastName_LostFocus()
    If Trim(txtLastName.Text) = "" Then
        sMsg = "You must enter your last name."
        Msgbox sMsg, vbExclamation, "Warning"
        txtLastName.SetFocus
    End If
End Sub
```

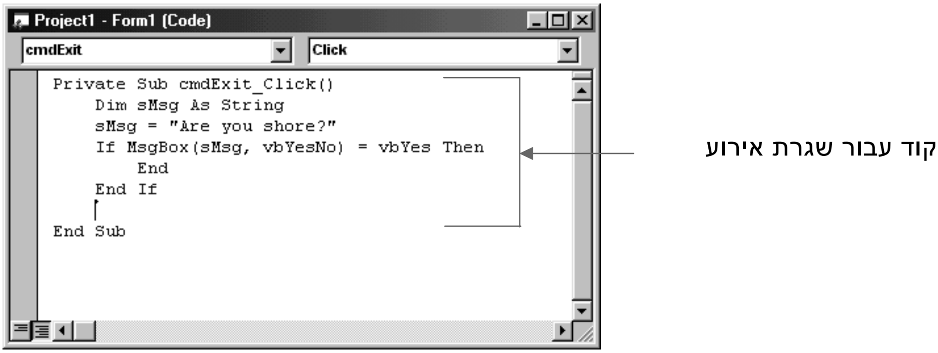
טיפול באירועים בתוכניות

בתוכנית טיפוסית יכולים לקרות הרבה אירועים המתרחשים כתוצאה מפעולות המשתמש. אירועים הקשורים לתוכנית יכולים להתרחש כתוצאה מפעולות כגון לחיצת המשתמש על לחצן העכבר, הקשה על מקש במקלדת, הזזת העכבר, שינוי הערך המופיע במאפיין הפקד, או מעבר לחלון אחר בתוכנית. התוכנית עצמה יכולה לגרום לאירועים להתרחש. אירועים הקשורים למערכת ההפעלה יכולים גם הם להתרחש כתוצאה מפונקציות של המערכת כגון השעון, או אפילו על ידי גורמים חיצוניים כמו קבלת דואר אלקטרוני. מאות, אם לא אלפי, אירועים יכולים להתרחש בכל תוכנית נתונה. מבנה התוכנית צריך להיות כזה שיוכל לבודד את האירועים הקשורים לפעולת התוכנית. Visual Basic מקלה על המלאכה הזאת.

קביעה מתי התרחש אירוע

כאשר Windows מזהה פעולה אשר התבצעה על ידי המשתמש והמכוונת לתוכנית שלנו, מערכת ההפעלה שולחת לתוכנית שלנו הודעה. מנגנון הריצה של Visual Basic המוטמע בתוכנית, מעבד את ההודעה ואם יש צורך, גורם לאירוע המתאים להתרחש. אם כתבנו שיגרה המטפלת בסוג האירוע האמור, מנגנון Visual Basic יפעיל את השיגרה האמורה.

כל אובייקט בתוכנית המבוססת על Visual Basic יכול להפעיל קבוצה מסוימת של אירועים לפי סוג האובייקט (חלון טקסט, טופס, לחצן, וכדומה). אולם, המפתח לטיפול נכון באירועים הינו לדעת בצורה מדויקת איזה שילוב של אובייקטים ואירועים נחוץ לתוכנית שלנו, וכן לכתוב את התוכנית לטיפול בשילובים אלה. ניתן לעשות זאת על ידי יצירת **שגרת אירוע** (Event Procedure) ב**חלון הקוד** (Code Window). לדוגמה, אם נרצה לכתוב תוכנית אשר תתבצע כאשר לחצן בשם **CmdExit** נלחץ, נכתוב את שגרת האירוע **Click** ללחצן **CmdExit**. ניתן לערוך את השיגרה על ידי פתיחת חלון קוד בטופס המכיל את **CmdExit**.



תרשים 5.2: Visual Basic תגיב לאירוע, רק אם נכתב עבורו קוד

בחלק **יצירת התוכנית הראשונה** מפרק 2, השתמשנו בתבניות מסוג DropDown בראש חלון הקוד כדי לבחור בשגרת האירוע המבוקשת. האובייקט נבחר בחלק השמאלי העליון של החלון (Object Box). שגרת האירוע הרצויה נבחרת בתיבה הימנית עליונה (Procedure Box). כאשר השילוב בין האובייקט לאירוע מתרחש בזמן ריצת התוכנית ובהנחה שנכתב קטע תוכנית המיועד לטיפול באירוע זה, מנגנון Visual Basic יריץ את קטע התוכנית המתאים. אם לא נכתב קטע תוכנית, Visual Basic תתעלם מהאירוע. תרשים 5.2 מראה דוגמה לשגרת אירוע אשר נכתבה במיוחד לאובייקט מסוים ולאירוע מוגדר.

סוגי אירועים

ישנם שני סוגים בסיסיים של אירועים אשר יכולים להתרחש בתוכנית המבוססת על Visual Basic: אירועים המתרחשים כתוצאה מפעולות המשתמש, ואירועים המתרחשים כתוצאה מפונקציה של המערכת. לרוב, נרצה לכתוב קטע קוד המטפל באירועים אשר המשתמש גורם להם. בצורה זו תהיה למשתמש שליטה על מהלך ריצת התוכנית. זאת אומרת שהמשתמש יכול לבצע פעולה מסוימת כאשר הוא ירצה (בהתחשב במבנה התוכנית), דבר המקנה למשתמש שליטה רבה.

הערה:

מובן שניתן להגביל את הפעולות שהמשתמש יכול לבצע על ידי הסתרת או חסימת אובייקטים כאשר לא נרצה שלמשתמש תהיה גישה אליהם. הטכניקה תוסבר בפרק 3.

ראה: שימוש במאפיינים לשליטה על התקשורת עם המשתמש.

אירועים המתרחשים כתוצאה מפעולת המשתמש - אירועים מסוג זה הינם אלה הקורים כתוצאה מפעולה שנעשתה על ידי המשתמש. כמו שבוודאי ניחשת, אירועים אלה כוללים הקשה על מקשי המקלדת ולחיצה על לחצני העכבר, אולם ישנם סוגים נוספים של אירועים אשר יכולים להתרחש כתוצאה מפעולת המשתמש בצורה ישירה או עקיפה. לדוגמה, כאשר המשתמש לוחץ על תיבת טקסט כדי לערוך את המידע הנמצא בתיבה, מתרחש אירוע Click עבור תיבת הטקסט. בנוסף לכך, ישנם עוד סוגי אירועים המתרחשים כתוצאה מהלחיצה.

אחד מהאירועים הללו הינו GotFocus המתרחש עבור תיבת הטקסט. אירוע זה מתרחש בכל פעם שהסמן נמצא באזור התיבה בין אם נלחץ לחצן העכבר או בשימוש במקש TAB. בנוסף, כאשר תיבת הטקסט נמצאת בפוקוס, אובייקט אחר חייב לאבד את הפוקוס. פעולה זו גורמת לאירוע LostFocus להתרחש עבור האובייקט השני. אירועים GotFocus ו-LostFocus מתרחשים כתוצאה מפעולות המשתמש, בדיוק כמו האירוע Click. כמו שנראה בחלק הבא **הבנת סדר האירועים**, ישנם מצבים שבהם קורה ריבוי אירועים כאשר המשתמש מבצע פעולה בודדת. בנוסף, חשוב להבין את הסדר שבו קורים האירועים.

הרשימה הבאה מכילה חלק מהפעולות הראשיות שעל ידן המשתמש יכול לגרום לאירועים להתקיים בתוכנית:

- הפעלת התוכנית.
- הקשה על מקש.
- לחיצה על לחצן העכבר.
- הזזת העכבר.
- סגירת התוכנית.

אירועים המתרחשים כתוצאה מפעולת המערכת - למרות העובדה שמסמכים ואובייקטים מסוגלים לטפל בסוגים רבים של אירועים, לרבים מהאובייקטים ישנם אירועים אפשריים משותפים, כפי שניתן לראות בטבלה 5.1.

טבלה 5.1: אירועים אופייניים לאובייקטים רבים

אירוע	מתרחש כאשר
Change	המשתמש משנה טקסט בתיבת טקסט או בתיבה משולבת.
Click	המשתמש לוחץ על אובייקט עם אחד מלחצני העכבר (שמאלי לרוב).
DblClick	המשתמש לוחץ לחיצה כפולה על אובייקט עם אחד מלחצני העכבר.
DragDrop	המשתמש גורר אובייקט למקום אחר.
DragOver	אובייקט נגרר מעל אובייקט.
GotFocus	אובייקט נמצא בפוקוס.
KeyDown	מקש הוקש כאשר אובייקט נמצא בפוקוס.
KeyPress	מקש הוקש ונעזב כאשר אובייקט נמצא בפוקוס.
KeyUp	מקש נעזב כאשר אובייקט נמצא בפוקוס.
LostFocus	הפוקוס הוסר מעל לאובייקט.
MouseDown	לחצן בעכבר נלחץ כאשר סמן העכבר נמצא מעל לאובייקט.
MouseMove	סמן העכבר זז מעל לאובייקט.
MouseUp	לחצן בעכבר נעזב בזמן שסמן העכבר נמצא מעל לאובייקט.

בוודאי שמת לב שכמה מהאירועים נראים כאילו הם מתייחסים לאותה פעולה של המשתמש. לדוגמה, האירועים Click, MouseDown ו-MouseUp קורים כאשר המשתמש לוחץ על לחצן העכבר. למראית עין כמה מההבדלים באירועים הינם מובנים מאליהם, לדוגמה אירוע MouseDown מתרחש כאשר לוחצים על לחצן העכבר, אכן ישנם הבדלים בין האירועים. במקרה של לחיצה על לחצן העכבר, אירוע Click במסגרת

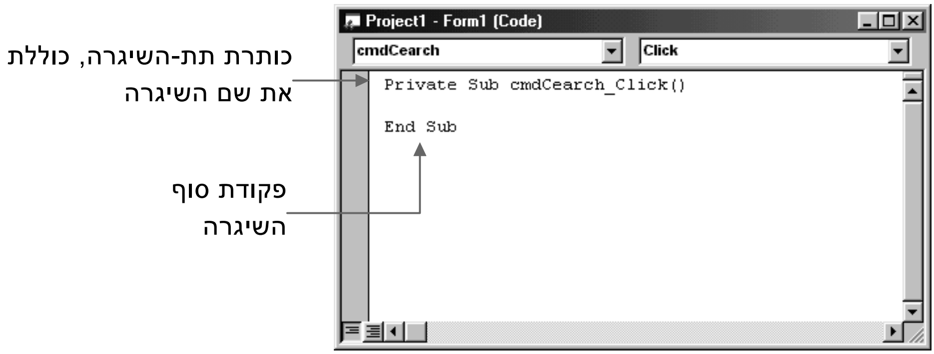
הפקדים ListBox, Check Box, Command Button או Option Button מתרחש אם נלחץ לחצן כלשהו בעכבר. האירוע אינו מתרחש במקרה שנלחץ לחצן אחר. האירועים MouseUp ו-MouseDown לא רק מתייחסים לכל לחצן עכבר שהוא, אלא שהאירוע מסוגל גם לדווח איזה לחצן נלחץ, כך שהתוכנית תוכל להגיב בהתאם.

האירועים KeyPress, KeyDown ו-KeyUp פועלים בצורה דומה. האירוע KeyPress מדווח איזה מקש הוקש בלבד ללא דיווח אם הוקש מקש אחר כגון Shift או Ctrl בזמן ההקשה על המקש המדווח. אם יש צורך במידע האמור, יש להשתמש באירועים האחרים.

כתיבת שגרות מונחות-אירועים

ובכן, כיצד נוכל לכתוב את התוכנית בסביבה מונחית אירועים? וכיצד ניתן לנפות את האירועים שאיננו מעוניינים להתייחס אליהם? התשובה לשתי השאלות זהה. כאשר כותבים את התוכנית, אנו כותבים שגרות המופעלות רק בתנאי שהתרחש אירוע מסוים באובייקט המתאים לשיגרה. כאשר אין שיגרה המיועדת לשילוב של אירוע ואובייקט מסוימים, Visual Basic תתעלם מהאירוע. השאלה הבאה היא כיצד לכתוב שיגרה המיועדת לאירוע מסוים?

כדי שנוכל לכתוב את התוכנית, יש צורך להגיע ל**חלון הקוד** (Code window). דבר זה נעשה על ידי לחיצה כפולה על אובייקט בטופס התוכנית (או על הטופס עצמו), על ידי בחירה באפשרות **Code View** בחלון הפרויקט, בלחיצה על תפריט **View** ובחירה באפשרות **Code**, או על ידי הקשה על F7. כל אחת מהאפשרויות הנזכרות מציגה את חלון הקוד המתאים לטופס הנוכחי (ראה תרשים 5.3).



תרשים 5.3: לחיצה כפולה על טופס או פקד במצב עיצוב מציגה את חלון הקוד, מוכן לשגרת האירוע המתאימה

לכתיבת תוכנית דוגמה, בצע את המהלכים הבאים :

1. הפעל את **Visual Basic** ובחר פרויקט סטנדרטי מסוג **EXE**.
2. לחץ לחיצה כפולה על הטופס כדי לפתוח את חלון הקוד. אם לטופס אין עדיין תוכניות, החלון ייפתח בשיגרה המקושרת לאירוע Load.
3. בחר באירוע **MouseMove** מתוך תיבת השגרות.

הערה:

שים לב שכאשר אנו בוחרים באירוע, Visual Basic בונה את שלד השיגרה המיוחסת לו בצורה אוטומטית. החלקים אשר מופיעים בשיגרה הם שם השיגרה ופקודת End sub אשר מצהירה על סוף השיגרה.

שם השיגרה המתייחסת לאירוע מסוים מכיל שם אובייקט ושם אירוע. בדוגמה, האובייקט הוא Form והאירוע MouseMove. שים לב ששגרת האירוע MouseMove מכילה בנוסף כמה משתנים. לדוגמה, המשתנים X ו-Y מייצגים את מיקום העכבר.

כעת תוכל לכתוב פקודות המסוגלות להגיב לכל שנרצה על אירוע שהתרחש. הוסף את השורות הבאות לשגרת האירוע :

```
Me.Caption = "Mouse Coordinates are (" & X & "," & Y & ")"
```

שגרת האירוע אמורה להיראות כזו :

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single,_
            Y As Single)
      Me.Caption = "Mouse Coordinates are (" & X & "," & Y & ")"
End Sub
```

הפקודות אשר הוקלדו לשיגרה מציגות את ערכי המשתנים X ו-Y בשורת הכותרת של הטופס על ידי קביעת המאפיין Caption של הטופס. לחץ על לחצן ההרצה של Visual Basic או הקש F5, והעבר את סמן העכבר מעל לחלון. כותרת החלון אמורה להשתנות כתוצאה מתזוזת העכבר.

טיפ:

אם נלחץ לחיצה כפולה על אובייקט, חלון הקוד שייפתח יציג את האירוע השכיח ביותר לאותו אובייקט. לדוגמה, אם נלחץ לחיצה כפולה על לחצן פקודה במצב עיצוב, חלון הקוד יציג את שגרת Click של הלחצן.

קריאה לשגרת קוד מתוך התוכנית

בתהליך למידת שפת התכנות, נצטרך לכתוב ולקרוא לשגרות רבות. שגרות מבוססות אירועים יוצאות דופן בכך שהן מופעלות על ידי מנגנון Visual Basic בעצמו. בדוגמת MouseMove בחלק הקודם, המנגנון הפעיל את הפקודה שכתבת בתגובה לאירוע שהתרחש, אולם כדאי לציין שתוכל להפעיל שגרות מבוססות אירועים בעצמך, בדיוק כמו כל שיגרה אחרת. תרצה לעשות זאת כאשר תכתוב תוכנית בתוך שגרת אירוע אשר אמורה להיות מופעלת במקומות נוספים.

הבה נדגים. אם חלון הפרויקט הקודם עדיין פתוח, חזור למצב עיצוב והצב לחצן פקודה בטופס. לחץ לחיצה כפולה על הלחצן ושנה את שגרת האירוע כך שתציג את התאריך והשעה הנוכחיים:

```
Private Sub Command1_Click()
       MsgBox "The system time is " & Now
End Sub
```

השורה שהוספת קלה להבנה. היא מציגה חלון הודעה כאשר לוחצים על הלחצן Command1. בנוסף, ניתן להפעיל את השיגרה Command1_Click מחלקים אחרים של התוכנית, ואפילו מתוך שגרת אירוע אחרת. לחץ לחיצה כפולה על הטופס, ובחר את אירוע Click מתוך תיבת השגרות.

שנה את שגרת Form_Click כך שתקרא לשגרת Command1_Click:

```
Private Sub Form_Click()
       Command1_Click
End Sub
```

הרץ את התוכנית ושים לב לכך שברגע שתלחץ על הטופס, מופיע אותו חלון ההודעה אשר מופיע כאשר תלחץ על לחצן הפקודה.

הערה:

אם שגרת האירוע שאנו קוראים לה דורשת ערכי משתנים, כמו בדוגמה בשגרת MouseMove, יש לספק אותם כאשר קוראים לשיגרה. בכל מקרה כשלפונקציה מוסיפים סוגריים היא אמורה להחזיר ערך.

הבנת סדר האירועים

כעת אתה יודע מהם אירועים וכיצד מנגנון Visual Basic מטפל בהם. ראית כיצד לכתוב פקודות המופעלות כאשר אירוע מתרחש, אבל עליך לצלול קצת יותר פנימה לתוך העולם המופעל על ידי אירועים.

כפי שכבר למדת, פעולת משתמש יחידה או אירוע הקשור במערכת, מסוגלים להפעיל סדרת אירועים. דבר זה יכול להועיל מפני שתוכל להשתמש באירועים שונים אלה כדי לטפל במצבים שונים, כמו במקרי MouseDown ו-Click. אולם בנוסף, יש לכך גם צד שלילי. אם תכתוב קוד יחיד למספר אירועים אפשריים, קוד זה עלול לפעול בצורה שלא תרצה שיפעל. במקרה הגרוע, רצף אירועים - כל אירוע והשיגרה שלו - עלולים לגרום למערכת להיכנס ללולאת תוכנה אינסופית. הצעדים הבאים מהווים נקיטת צעדים המיועדים למנוע בעיות אלה:

- זיהוי האפשרות שמספר אירועים יתרחשו.
- זיהוי איזה אירועים יתרחשו כאשר המשתמש מבצע פעולה מסוימת.
- הבנת סדר התרחשות האירועים.
- בדיקת הקשר בין אירועים שונים העלולים לפעול בו זמנית.

בחלקים הבאים נבדוק נושאים שיש לשים אליהם לב בזמן כתיבת שגרות אירוע ב- Visual Basic.

ריבוי אירועים בפעולה יחידה

אחת הבעיות בטיפול בריבוי אירועים היא שהמשתמש יכול לתקשר עם התוכנית בהרבה צורות. לדוגמה, האם הוא הגיע לתיבת הטקסט בעזרת העכבר או בעזרת מקש TAB? לפני התנועה לתיבה, האם היה פוקוס על תיבת טקסט אחרת או על לחצן פקודה? ומה קורה כאשר עוברים מטופס אחד למשנהו? כפי שתראה, ריבוי האפשרויות עלול לגרום לבלבול.

לעומת זאת, החדשות הטובות הן שלא תכתוב קוד לרוב צירופי האירועים/ אובייקטים. לכן, למרות העובדה שהאירועים הנזכרים עלולים לקרות, התוכנית שתכתוב תתעלם מהם ולפיכך הם לא יצרו שום בעיה.

למרות העובדה שאי כתיבת הקוד המיותר הופכת את מלאכת הטיפול בריבוי אירועים לפשוטה יותר, צריך עדיין לבצע עבודה מסוימת. כדי לטפל במצב זה, הבה ונסתכל על כמה סדרות אירועים אשר קורות כאשר המשתמש מבצע פעולה.

קח בחשבון, לדוגמה, את תיבת הדו-שיח המיועדת לכניסת המשתמש ל-Windows. אם נתאר לנו שתיבה זו נכתבה ב- Visual Basic, אזי שבזמן הקלדת שם המשתמש והסיסמה קורים כמה אירועים.

קודם כל, הבה נבחן הקשת מקש בודדת. לכל תו שנקליד בשם המשתמש מתרחשים שלושה אירועים: KeyPress, KeyDown ו-KeyUp (בסדר זה). כאשר אנו מזיזים את

הסמן משדה שם המשתמש אל שדה הסיסמה, אנו מעבירים את הפוקוס מאובייקט אחד למשנהו, כך שמתרחשים שני אירועים: LostFocus עבור האובייקט הנוכחי, ו-GotFocus עבור האובייקט החדש.

לבסוף, כאשר לוחצים על לחצן OK, לחיצת העכבר הפשוטה יוצרת שלושה אירועים: MouseUp, MouseDown ו-Click. אם נלחץ לחיצה כפולה על אובייקט אחר, כגון הטופס, יווצרו שני אירועים נוספים לאחר האירוע Click: DblClick ו-MouseUp נוסף. סך הכל יש כבר חמישה אירועים הקשורים למה שנראה כפעולה יחידה של המשתמש. בנוסף, אם לחיצת העכבר גורמת לפוקוס לעבור מאובייקט אחד לשני, מתרחשים גם אירועים LostFocus ו-GotFocus, סך הכל שבעה אירועים.

העובדה שניתן לבצע פעולות שונות כדי להגיע לאותה מטרה מהווה בעיה נוספת. לדוגמה, בואו נבחן את הלחצן OK בתיבת זיהוי המשתמש של Windows. האם ידעת שניתן להפעיל אותו על ידי לחיצה על העכבר **וגם** על ידי מעבר אליו עם מקש Tab והקשה על מקש הרווח? השימוש בעכבר לביצוע המשימה מפעיל את שגרות MouseDown, MouseUp ו-Click, בעוד ששימוש במקלדת מפעיל את שגרות KeyDown, KeyPress, KeyUp ולבסוף את שגרת Click. תהליך מבלבל, הלא כן?

ואולם, ניתן לדעת איזה אירועים יקרו בזמן ריצת התוכנית ובאיזה סדר הם יקרו.

מציאת סדר התרחשות האירועים

המפתח להבנת מהות האירועים הינו הידיעה **מתי** הם קורים. אם מידע זה מעורפל עדיין, נסה את הדוגמה הבאה. בדוגמה זו, תיצור תוכנית הבודקת אירועים הקשורים בטופס ומפעילה את שגרות האירועים המתרחשים כדי לדעת מה סדר התרחשותם. ניתן גם לצפות בסוג הקוד שתרצה לשלב בכל אחת משגרות האירועים הבאות:

יצירת תוכנית דוגמה המטפלת באירועים הקשורים לטופס. יש חמישה אירועים מיוחדים לכל טופס, ותוכל לשלב קוד תוכנית בשגרות האירועים של כל אחד מהם:

- **Load**. מתרחש כאשר הטופס מועלה לזיכרון.
- **Activate**. מתרחש כאשר הטופס מוצג לראשונה, או כאשר המשתמש חוזר אליו מטופס אחר.
- **Deactivate**. מתרחש כאשר המשתמש עובר לטופס אחר או כאשר הטופס נסתר.
- **Unload**. מתרחש כאשר הטופס מוסר מהזיכרון.
- **Initialize**. מתרחש כאשר נוצר עותק של הטופס.
- **Terminate**. מתרחש כאשר עותק של הטופס נמחק.

תוכנית הדוגמה פשוטה מאוד. יש בה שני טפסים בלבד ללא אובייקטים. כדי ליצור אותה, בצע את הפעולות הבאות:

1. פתח את **Visual Basic** וצור פרויקט רגיל מסוג **EXE**.
2. לחץ לחיצה כפולה על **Form1** כדי להציג את חלון הקוד שלו.

3. ודא ששגרת **Load** נבחרה בתיבת האירועים בחלק הימני-עליון של חלון הקוד.

4. הקלד את הפקודה **MsgBox "The Form Load Event Occurred"** לתוך שגרת האירוע. השיגרה אמורה להיראות כך:

```
Private Sub Form_Load()
      MsgBox "The Form Load Event Occurred"
End Sub
```

5. חזור על שלבים 3 ו-4 כדי להציב פקודות **MsgBox** דומות בשגרות **Initialize** ו-**Terminate1 Unload**.

6. הוסף טופס שני בשם **Form2** לפרויקט על ידי בחירת האפשרות **Add** מתוך תפריט **Project**.

כעת, כשהפרויקט מוכן, הקש F5 כדי להריץ אותו. שים לב ששגרת Initialize מופיעה בהתחלה, ואחריה שגרת Load. לחץ על לחצן הסגירה של הטופס, וצפה בהפעלת שגרות Unload ו-Terminate לפני שהתוכנית מפסיקה לפעול.

בדיקת אירועי התוכנית. כעת הרץ את התוכנית מחדש ולחץ אישור להודעות Initialize ו-Load, לאחר שהטופס עולה הקש Ctrl+Break כדי להשהות את המשך הריצה. הקש Ctrl+G כדי להביא את החלון המיידי (Immediate). הצב את הסמן בחלון זה והקלד את השורות הבאות (הקש ENTER לאחר כל שורה) תוך שימת לב להודעות המופיעות:

```
Load Form2
Unload Form1
Load Form1
```

אנו מעלים את Form2 כדי למנוע מהתוכנית להסתיים כאשר מסירים את Form1. שים לב לכך שאירוע Initialize לא התרחש כאשר Form1 עלה בפעם השנייה. כעת הקלד את השורות הבאות:

```
Unload Form1
Set Form1 = Nothing
```

קביעת ערך הטופס ל-Nothing הינה הרגל טוב. בצורה זו ניתן להבטיח שכל המשאבים אשר הוקצו לטופס שוחררו בצורה נאותה.

כעת הקלד את השורה הבאה:

```
Load Form1
```

אירוע Initialize התרחש בשנית מפני שהעותק הקודם של Form1 אשר יצרת לפני כן נמחק כאשר קבעת אותו כ-Nothing.

שימוש באירועי טופס. ניתן להשתמש בקוד תוכנית באירועים אלה כדי להגדיר את מאפייני הטופס או אחד מהאובייקטים שלו, הגדרת בסיסי נתונים או מסדי מידע הדרושים לטופס, או הפעלת קוד אחר שתרצה בו. האירועים Load ו-Unload מתרחשים כל אחד רק פעם אחת במשך זמן החיים של טופס - כאשר הטופס נטען לזיכרון וכאשר

הוא מוסר ממנו, בסדר זה. מצד שני, האירועים Activate ו-Deactivate יכולים להתרחש פעמים רבות. לכן, יש צורך בקביעה זהירה של קטעי הקוד בשגרות המתאימות.

קטע הקוד הבא מציג כמה דברים פשוטים אך שימושיים שאפשר לעשות עם קוד בשגרת Load. שימוש נפוץ הוא לאתחל ערכי מאפיינים של אובייקטים בטופס (או את מאפייני הטופס עצמו). תוכנית 5.2 מדגימה כיצד להגדיר את השדה Caption של הטופס בעת טעינת הטופס, כדי להציג את שם המשתמש הנוכחי (שאנו מניחים שהוזן לפני כן במקום אחר בתוכנית ונשמר במשתנה מחרוזת בשם sUserName). בנוסף, תראה כיצד להגדיר את שדה Text של תיבת טקסט כך שתכיל את התאריך הנוכחי.

תוכנית 5.2: TestLoad.TXT – **שימוש בשגרת** Load **לצורך הגדרת מאפיינים.**

```
Private Sub Form_Load()
	Me.Caption = "Expense Summary for " & sUserName
	txtDate.Text = Date
End Sub
```

שימושים נפוצים אחרים בשיגרה Load כוללים יצירת חיבורים למידע השמור במאגרי מידע, אתחול משתנים ומיקום הטופס בצורה יחסית לטפסים אחרים בתוכנה אשר ייפתחו לאחר מכן.

שגרת אירוע שימושית נוספת הקשורה לטפסים הינה השיגרה Resize. היא מופעלת בזמן שגודל הטופס משתנה, בין אם על ידי המשתמש או על ידי התוכנית. האירוע מתרחש גם לאחר אירוע Load ולפני אירוע Activate. לרוב משתמשים בשגרת Resize כדי לשנות את גודל האובייקטים בטופס. בצורה זו נוצר למשתמש יותר מקום לעבוד בו כאשר שטח הטופס גדל ומונע ממידע חיוני להיעלם מהעין כאשר שטחו קטן. תוכנית 5.3 מראה כיצד גודל רשת הרקע בטופס (Grid) משתנה, כאשר גודל הטופס משתנה. בנוסף, התוכנית בודקת את המאפיין Windowstate של הטופס ואינה מבצעת את הפעולה כאשר הטופס במצב ממוזער.

תוכנית 5.3: Resize.TXT – **שינוי גודל ומיקום אובייקטים כאשר גודל הטופס משתנה.**

```
If Me.Windowstate <> vbMinimized And AllowResize Then
	dbgResults.Width = Me.ScaleWidth – 180
	dbgResults.Height = Me.ScaleHeight – dbgResults.Top – 60
End If
```

הצעד הראשון בקביעת פעולת השיגרה הינו מיפוי סדר התרחשות האירועים בתוכנית. נוכל לכתוב תוכנית פשוטה כדי לתרגל מצבים שבהם מתרחשים אירועים. בתחילה ניצור פרויקט חדש סטנדרטי מסוג EXE ונמקם עליו TextBox ולחצן. אחר נכניס פקודות מתאימות לשגרות האובייקט והטופס כדי לדעת מתי מתרחש אירוע. הדרך הפשוטה לעשות זאת היא להשתמש בפקודת Debug.Print אשר גורמת להופעת טקסט בחלון Immediate. הוסף את פקודת Debug.Print לשגרות האירוע בתיבת הטקסט.

```
Private Sub Text1_Click()
	Debug.Print "Text1 Click Event"
End Sub
Private Sub Text1_GotFocus()
	Debug.Print "Text1 Got Focus Event"
End Sub
Private Sub Text1_KeyPress(KeyAscii As Integer)
	Debug.Print "Text1 KeyPress Event"
End Sub
```

בפעמים שתשתמש בשיטת קבלת מידע כזו, יש חשיבות עצומה להבנת סדר התרחשות האירועים. ברגע שתבין כי פעולה בודדת של משתמש יכולה להוביל להתרחשות מספר אירועים, תוכל להשתמש בידע זה כדי לתכנן את הקוד שתרצה לכתוב לכל שיגרה. לדוגמה, ניתן להשתמש בשיגרה הראשונה המתבצעת בתיבת טקסט, כאשר המשתמש מקליד אות בזמן שהתיבה נמצאת בפוקוס, KeyDown, כדי לדעת אם המשתמש הקיש על מקש Ctrl. בדרך זו ניתן לדעת בתוכנית אם התבצעה פעולה מיוחדת כגון פתיחת רשומה חדשה או שמירת מידע הקשור לאתחול התוכנית. בכל אחד מהמקרים, תיאלץ לבדוק את התוכנית בצורה יסודית תוך התחשבות במיגוון רב של מהלכים אפשריים באובייקטים, כדי לוודא שלא יתרחשו מהלכים לא צפויים אשר יכולים לגרום לתוצאות לא רצויות.

מכאן...

פרק זה התמקד הצורה שבה Visual Basic מטפלת באירועים. אולם אירועים הינם שימושיים רק במקרים שבהם משתמשים בטפסים ואובייקטים מפני שאלה אובייקטים המסוגלים לקלוט התרחשות אירועים. בנוסף, התרחשות אירועים עלולה לגרום לפעולה בתוכנית רק במידה ונכתבה שיגרה המיועדת לטפל באירוע. ניתן לקבל מידע נוסף על הנושאים הקשורים בפרקים הבאים:

- ראה פרק 3 **אבני היסוד של Visual Basic** לקבלת הנחיות לגבי טפסים, אובייקטים, מאפיינים, שיטות ואירועים.
- למידע נוסף הקשור בשימוש באובייקטים, ראה פרק 4, **שימוש בפקדי ברירת המחדל של Visual Basic**.
- פרק 6 **שליטה נוספת למשתמש: תפריטים וסרגלי כלים** מכסה את חלק התפריטים וסרגלי הכלים בתכנות ב- Visual Basic.
- למידע אודות פקדי הדו-שיח ראה פרק 7 **שימוש בתיבות דו-שיח לקבלת מידע**.
- ראה פרק 11 **ניהול הפרויקט: תת-שגרות, פונקציות וריבוי טפסים** כדי ללמוד יותר את נושא כתיבת השגרות הפרטיות.
- כדי ללמוד על סוגי האירועים הרבים הקיימים באובייקטים המיוחדים, ראה פרק 12 **הפקדים המשותפים של Microsoft**.

6

שליטה נוספת למשתמש: תפריטים וסרגלי כלים

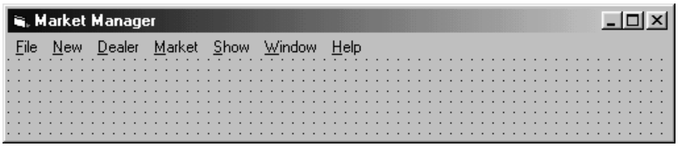
מה בפרק?

- יצירת שורת תפריטים
- יצירת תפריטים מקוצרים (Pop-up Menus)
- שימוש בסרגלי כלים ב- Visual Basic
- שימוש בפקד סרגל הכלים המשופר CoolBar

תוכניות רבות הינן דו-כיווניות, משמע שהן מבצעות פעולות כתגובה לפעולות המשתמש. בפרק 5 **תגובה באמצעות שגרות אירוע**, למדת כיצד לגרום לאובייקטים להגיב לפעולות המשתמש. בפרק זה, נבדוק כיצד לתת למשתמש שליטה רחבה יותר על הנעשה. תלמד כיצד להוסיף מבני תפריטים לתוכניות, תוך מתן יכולת למשתמש לבצע בחירה מתוך מספר אפשרויות בתוכנית. אחר תלמד כיצד להוסיף סרגלי כלים כדי לאפשר למשתמשים קיצורי דרך לפונקציות שכיחות בתוכנית.

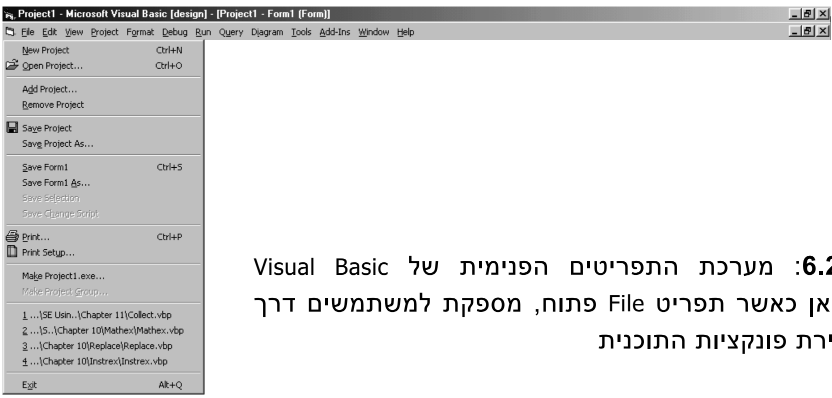
יצירת שורת תפריטים

אחד הצדדים החשובים בעיצוב תוכנית, הוא לאפשר למשתמש גישה קלה ונוחה לפונקציות התוכנית. המשתמשים רגילים להגיע לרוב הפונקציות בעזרת לחיצה או שתיים על העכבר. בנוסף הם מעוניינים שכל הפונקציות תהיינה ממוקמות בצורה נוחה במקום אחד. כדי להקל על ביצוע המשימה, Visual Basic מאפשרת לנו ליצור מערכת תפריטים בצורה מהירה ופשוטה בעזרת עורך התפריטים. ניתן להשתמש בעורך כדי ליצור שורת תפריטים לכל אחד מטפסי התוכנית וכן לתפריטים גמישים אשר מופיעים בהתאם למיקום הסמן (ראה סעיף **יצירת תפריטים מקוצרים** בהמשך הפרק). שורת התפריטים של הטופס הראשי בתוכנית לניהול תערוכה מוצגת בתרשים 6.1.



תרשים 6.1: מערכת תפריטים מעוצבת בצורה טובה מאפשרת למשתמשים למצוא ולהפעיל את הפונקציות בצורה נוחה

הצעד הראשון ביצירת תפריט הוא קביעת הפקודות הנכללות בו וסדר הופעתם. תרשים 6.2 מציג את מערכת התפריטים הפנימית של Visual Basic. כפי שניתן לראות, הפקודות מסודרות בקבוצות פונקציונליות (קובץ, עריכה, תצוגה וכיוצא בזה).



תרשים 6.2: מערכת התפריטים הפנימית של Visual Basic המוצגת כאן כאשר תפריט File פתוח, מספקת למשתמשים דרך נוחה לבחירת פונקציות התוכנית

החלקים הבאים מספקים מידע על בנייה ותכנות תפריטים.

תפריטים נפוצים

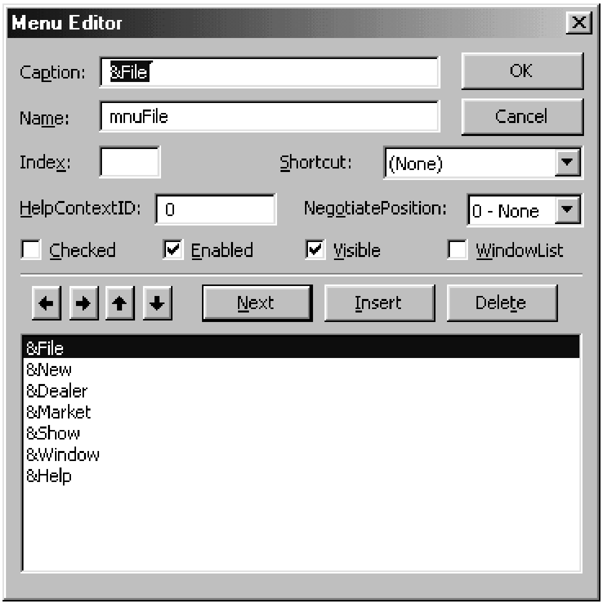
כאשר אנו יוצרים תפריט בתוכנית, אנו צריכים לאחד מרכיבים דומים. למעשה, במידה והדבר אפשרי, נוכל להשתמש בקבוצות המוכרות למשתמש. בדרך זו, יש למשתמש מושג היכן למצוא אפשרויות מסוימות בתפריט, אפילו במידה והוא לא השתמש בתוכנית לפני כן. הרשימה הבאה מתארת מספר תפריטים אופייניים שניתן למצוא אותם בשורות תפריטים רבות:

- קובץ **File**. תפריט זה מכיל את הפונקציות הקשורות בטיפול בקבצים הנמצאים בשימוש על ידי התוכנית. חלק מהאפשרויות הנפוצות הן **חדש** (New), **פתיחה** (Open), **סגור** (Close), **שמור** (Save), **שמירה בשם** (Save As) **והדפסה** (Print). אם התוכנית עובדת עם קבצים שונים בצורה רציפה, נוכל להוסיף לתפריט גם רשימה של הקבצים האחרונים שהתוכנית השתמשה בהם. בסוף תפריט זה נהוג להציב גם את פקודת **היציאה** (Exit) מהתוכנית.
- עריכה **Edit**. תפריט העריכה מכיל את הפונקציות הקשורות לעריכת טקסט ושימוש בלוח (Clipboard) הפנימי של Windows. חלק מהאפשרויות הנפוצות הן **בטל** (Undo), **גזור** (Cut), **העתק** (Copy), **הדבק** (Paste), **נקה** (Clear), **חיפוש** (Find) **והחלפה** (Replace).
- תצוגה **View**. ניתן לכלול את התפריט אם יש אפשרות להציג את הטופס בתוכנית במספר צורות. לדוגמה, מעבד תמלילים יכול לכלול אפשרות לתצוגה רגילה בעריכת הטקסט ותצוגה לפי דפים למיקום מרכיבי הטופס, וכן מיגוון אפשרויות הגדלה (Zoom). שימוש נוסף בתפריט **תצוגה** הוא לאפשר למשתמשים להציג או להחביא טפסים מיוחדים בתוכנית (כגון האפשרות ToolBox בתפריט View).
- כלים **Tools**. בתפריט ניתן למצוא את מכלול פונקציות העזר והפונקציות הכלליות. לדוגמה, ניתן לשלב בתפריט זה בודק שגיאות כתיב, בודק תחביר, או עורך משוואות בתוכנית עיבוד תמלילים.
- חלון **Window.** תפריט זה נכלל בדרך כלל במקרים בהם התוכנית משתמשת בממשק ריבוי טפסים (MDI). תוכניות מסוג MDI, כגון מעבד התמלילים Word, תומכות בעריכה של מספר טפסים באותו מופע של Word (כלומר תחת תוכנית Word אחת). תפריט **חלון** מאפשר למשתמש לארגן את תצוגת הטפסים הפתוחים, או להחליף ביניהם בצורה נוחה.
- עזרה **Help**. תפריט העזרה מכיל גישה למערכת העזרה של התוכנית. לרוב, תפריט זה מכיל אפשרויות לאינדקס עזרה (תוכן עניינים), אפשרות חיפוש (כדי לאפשר למשתמש למצוא נושא מסוים בצורה מהירה) ומידע אודות התוכנית (מידע כללי, זכויות יוצרים ושמות כותבי התוכנית).

ניתן להשתמש בששת התפריטים האופייניים הללו כבסיס ליצירת מערכת התפריטים הפרטית שלנו. השתמש באחד או יותר מתוכם במידת הצורך והוסף קבוצות תפריטים נוספות לפי הדרישה, אולם תכנן היטב את התפריטים הנוספים לפני ההוספה. ודא שהמשתמשים יוכלו להגיע בקלות לכל הפונקציות דרך מערכת התפריטים.

הגדרת המרכיבים העיקריים

לאחר שהחלטנו איזה פונקציות לכלול בתפריטים וכיצד לחלק את הפונקציות הללו לקבוצות, אפשר להתחיל לבנות את התפריט. כדי ליצור תפריט, יש לפתוח תחילה את הטופס שבו רוצים ליצור אותו ואז להפעיל את עורך התפריטים באחת משלוש הדרכים הבאות: לחיצה על לחצן **Menu Editor** (עורך התפריטים) בסרגל הכלים, בחירת תפריט **Tools** ואפשרות **Menu editor**, או הקשה על Ctrl+E. פעולות אלו תגרומנה להופעת עורך התפריטים, כמתואר בתרשים 6.3.



תרשים 6.3: עורך התפריטים מאפשר דרך קלה ליצור תפריטים לתוכניות

פקדי תפריט

כל שורת טקסט (אפשרות בתפריט) נקראת **פקד תפריט** (Menu Control), בדיוק כשם שלחצן פקודה שאנו מציבים בטופס נקרא **פקד לחצן** (CommandButton Control). אנו נשתמש בעורך התפריטים לצורך יצירת פקדים בתפריט והגדרת המאפיינים הבאים לכל פקד:

- **Caption** (כותרת). זהו הטקסט אשר יציג את הפקד בתפריט. ניתן להגדיר מקש קיצור לבחירת הפקד מתוך התפריט, על ידי הוספת הסימן "&" לפני האות שנבחרה להפעלת קיצור הדרך. לדוגמה, אם המאפיין Caption של פקד מסוים בתפריט מכיל את המחרוזת F&ormat, לאות O יתווסף קו תחתון והמשתמש יוכל לבחור את הפקודה בהקשה על מקש O במקלדת, כאשר התפריט פעיל. יש להגדיר מאפיין Caption עבור כל הפקדים בתפריט.
- **Name**. משתמשים במאפיין זה כדי לזהות את האפשרות בתפריט בקוד התוכנית. יש להגדיר מאפיין Name עבור כל האפשרויות בתוכנית. בדומה לפקדים

האחרים, לכל פקד בתפריט חייב להיות שם אשר בעזרתו תוכל התוכנה לזהות אותו. בשונה מפקדים אחרים, Visual Basic אינה מצמידה לפקדי התוכנית שם באופן אוטומטי ולכן אתה חייב לציין שם עבור כל פקד לפני סגירת עורך התפריטים (ניתן לראות במאפיין name שם פנימי של הפקד בתוך התוכנית ואילו המאפיין caption הוא שם חיצוני עבור המשתמש).

- **Inde<u>x</u>**. אם הפקד המוגדר הוא חלק ממערך פקדים, מאפיין Index משמש לצורך זיהוי הפריט בתוך המערך.
- **<u>S</u>hortcut**. בעזרת מאפיין זה ניתן להגדיר צירופים של מקשי קיצור שיאפשרו למשתמש לבחור פקד מסוים בתפריט בעזרת צירוף מקשים, וכך לעקוף את מערכת התפריטים. לדוגמה, תוכניות רבות משתמשות בצירוף Ctrl+P כקיצור ומבטלות את הצורך לבחור את האפשרות Print מתוך תפריט File במערכת התפריטים.
- **<u>H</u>elpContextID**. מאפיין זה מגדיר שדה מזהה המאפשר עבודה מול קובץ עזרה מותאם אישית ומאפשר מתן עזרה רגישת הקשר (Context Sensitive Help) לתוכניתך.
- **Neg<u>o</u>tiatePosition**. אם ביישום שלנו יש אובייקט המקושר למקום אחר או מוטמע בתוכנית, מאפיין זה קובע אם וכיצד יופיע פקד זה בזמן שהאובייקט פעיל.
- **<u>C</u>hecked**. אם ערך מאפיין מוגדר True, יופיע סימון (Check mark) משמאל לפקד בתפריט. בדרך זו ניתן לסמן, לדוגמה, כי המשתמש בחר להפעיל אפשרות מסוימת. אם ערך המאפיין הוא False, לא יופיע סימון.
- **<u>E</u>nabled**. ניתן להגדיר מאפיין זה כ-False במידה ואנו מעוניינים לנטרל את פקד התפריט בזמן מסוים. לדוגמה, אם לא מסומן כל טקסט שהוא, אפשר לחסום את האפשרות Copy בתפריט Edit על ידי קביעת ערך המאפיין כ-False.
- **<u>V</u>isible**. קובע האם ניתן יהיה לראות את הפקד בתפריט. אפשר לכלול בתוכנית פקדי תפריט ולגרום לכך שלא ייראו על המסך במצבים מסוימים. לדוגמה, לא נרצה שהאפשרות Window תופיע בתפריט כאשר אין חלונות פתוחים.
- **<u>W</u>indowList**. מאפיין זה מאפשר הוספת רשימה דינאמית של כל חלונות המשנה הפעילים, לתפריט רמה עליונה ביישום MDI.

יצירת מערכת פשוטה של תפריטים

נדגים את תהליך עיצוב התפריטים על ידי יצירה של מערכת תפריטים פשוטה. ניצור מבנה תפריט הכולל מספר פריטים סטנדרטיים. ראשית, צור תפריט File לדוגמה:

1. הפעל את Visual Basic והתחל פרויקט רגיל מסוג EXE.
2. ודא שהטופס שתרצה ליצור בו את התפריט מסומן. הטופס Form1 אמור להופיע על המסך כשהוא מסומן. אם לא, בחר אותו.

3. הפעל את עורך התפריטים על ידי לחיצה על **Tools, Menu Editor** או על ידי לחיצה על הלחצן **Menu Editor** בסרגל הכלים. הסמן אמור להופיע במאפיין Caption. מאפיין זה מכיל את הטקסט המזהה את הפקד בתפריט הנוכחי.

4. הקלד **&File**. זכור כי הסימן "&" מציין כי התו הבא (במקרה זה F), ישמש כמקש הגישה לפקד. בזמן ההקשה, שים לב לכך שהטקסט המוקלד מופיע בתיבה שבתחתית עורך התפריטים בצורה אוטומטית. באזור זה ניתן לראות את התפריט בצורה היררכית. כמו כן, שים לב כי השורה העליונה בתיבה מוארת.

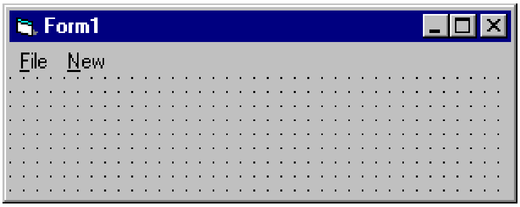
5. עבור למאפיין **Name** בעזרת טאב והקלד **mnuFile**. שני המאפיינים שהגדרת - Caption ו-Name הם כל שתצטרך להגדיר לעת עתה. ברירות המחדל של המאפיינים האחרים מספיקות לשם דוגמה זו.

6. לחץ על לחצן **Next** או הקש על **Enter** כדי לקבל את המאפיינים שהגדרת לפריט זה. תיבות המאפיינים תנוקנה ותהיינה מוכנות לפריט הבא. שים לב שהחלק המואר בתיבה בחלק התחתון של העורך עוברת לשורה הבאה (שורה ריקה).

7. חזור על צעדים 4 עד 6 כדי ליצור פריט שני. השתמש ב-**&New** במאפיין **Caption** ו- **mnuFileNew** כשם.

8. לחץ **OK** כדי לסגור את עורך התפריטים.

לאחר ביצוע הצעדים הנזכרים, אמורה להיות שורת התפריטים בטופס כמו זה שבתרשים 6.4.



תרשים 6.4: שורת תפריטים זו, הנראית כאן במצב עיצוב, נוצרה בעזרת עורך התפריטים

כמובן, שהתפריט אמור להיראות בצורה שונה, כאשר **New** הינו תפריט משנה של **File** ולא תפריט בפני עצמו. כדי לפתור את הבעיה, תצטרך להתוודע להבדלי הרמות האפשריות בתפריט. הבדלים אלה תכיר בסעיף הבא, **תפריטים בעלי מספר רמות**.

כמו כן, במידה וניסית ללחוץ על אחד מהפריטים בסביבת העיצוב, הופיע לפניך חלון הקוד של שיגרה. למעשה, הפעולה שביצעת, דומה ללחיצה כפולה על אובייקט בטופס בזמן עיצוב: פתחת את שגרת Click של הפריט. כאן תוכל לכתוב קוד אשר יתבצע במידה והפריט נבחר בזמן ריצת התוכנית. תוכל ללמוד אודות כתיבת קוד לפריטי תפריטים מאוחר יותר בפרק זה בחלק **כתיבת קוד לפריטי תפריטים**. לעת עתה, סגור את חלון הקוד.

תפריטים בעלי מספר רמות

התפריטים האופייניים ביישומים המבוססים על Windows, מכילים מספר רמות של פקודות. הפריטים ברמה הגבוהה ביותר הינם אלה המופיעים בשורת התפריטים של התוכנית. בדרך כלל, לחיצה על פריט שכזה פותחת את תפריט המשנה המשויך לפריט הראשי. כל פריט בתפריט המשנה יכול לייצג פונקציה בתוכנית או לגרום להופעת תפריט משנה נוסף. ב- Visual Basic, ניתן ליצור עד שש רמות של תפריטים.

טיפ:

למרות שניתן ליצור עד שש רמות של תפריטים ב- Visual Basic, כדאי בדרך כלל להגביל את התפריטים לשתים עד שלוש רמות. יותר מדי רמות עלולות להקשות על המשתמש לנווט את דרכו בתפריט.

בצע את המהלכים הבאים כדי להמשיך בהכנת תפריט **File** אשר יצרת קודם לכן ובנוסף צור גם תפריטי **Edit** ו-**Options**.

1. סמן את **Form1** ופתח את עורך התפריטים.
2. בתיבה בתחתית עורך התפריטים, בחר את פריט **New**.
3. זהה את ארבעת מקשי החיצים בחלק האמצעי-שמאלי של העורך והקש על החץ הפונה ימינה פעם אחת. פעולה זו מורידה את הפריט רמה אחת למטה. שים לב שתוכן הפריט מוזז פנימה כסימן לכך שהפריט עצמו נמצא ברמה נמוכה יותר. לחיצה על החץ הפונה שמאלה מזיזה את הפריט רמה אחת למעלה.
4. לחץ על לחצן **Next** כדי להכין את עורך התפריטים לפריט חדש. שים לב לכך שבנוסף לניקוי תיבות המאפיינים, הפריט החדש והריק מוזז באופן אוטומטי לרמה בה מצוי הפריט הקודם.
5. הוסף פריטים חדשים באותה הרמה, ובהם הערכים הבאים :

Caption	**Name**
&Open	mnuFileOpen
&Print	mnuFilePrint
&Save	mnuFileSave
Sen&d To	mnuFileSendTo

6. פריט **Send To** יהפוך בעצמו להיות תפריט משנה. לאחר יצירתו, הזז את הפריט הריק הבא רמה אחת למטה על ידי לחיצה על החץ הפונה ימינה והוספת פריטים חדשים (המוזזים פעמיים) לתפריט :

Caption	**Name**
&Mail Recipient	mnuFileSendToMail
&Fax Recipient	mnuFileSendToFax

הערה:

כאשר אנו יוצרים תפריטים בעלי מספר רמות כמו באפשרות Send To בתפריט הדוגמה, Windows משלבת במקום המתאים בתפריט, סימן דמוי חץ המאפשר למשתמש לדעת שלאפשרות זו מסונף תפריט משנה.

7. לחץ על החץ הפונה שמאלה פעם אחת כדי להעביר את הפריט הבא בתפריט לרמה גבוהה יותר. הגדר את מאפיין Caption שלו כ-**E&xit** ואת שמו כ-**mnuFileExit**.

8. לחץ על **Next** או הקש **Enter** כדי ליצור פריט חדש.

9. לחץ על החץ הפונה שמאלה כדי להגדיר את הפריט כתפריט ברמה העליונה (ללא הזזה). הגדר את מאפיין **Caption** כ-**&Edit** ואת שמו כ-**mnuEdit**.

10. הזז את הפריט הבא בתפריט כך שהוא יופיע כתת פריט בתפריט **Edit**. הוסף את הפריטים הבאים:

Caption	**Name**
Cu&t	mnuEditCut
&Copy	mnuEditCopy
&Paste	mnuEditPaste

11. הוסף תפריט ראשי נוסף בעצמך. הגדר את מאפיין **Caption** כ-**&Options** ואת שדה השם כ-**mnuOptions**.

12. הוסף את הפריטים הבאים תחת תפריט **Options**:

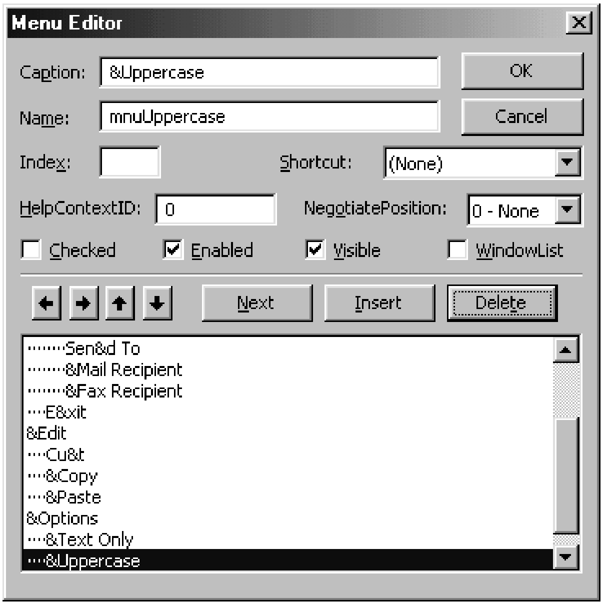
Caption	**Name**
&Text Only	mnuOptionsText
&Uppercase	mnuOptionsUppercase

עורך התפריטים אמור להיראות כעת כמו בתרשים 6.5.

13. הקש **OK** כדי לסגור את עורך התפריטים.

טיפ:

מומלץ לתת לפקדי התפריטים (כלומר הפריטים השונים בתפריט) שמות הירככיים. באופן זה, שמו של כל פקד בתפריט מתחיל בקידומת mnu ולאחריו מילה המתארת את החלק בתפריט שבו הפריט נמצא. לדוגמה, שם הפריט אשר דרכו מעבירים מידע לפקס הינו mnuFileSendToFax. שמו של הפקד מרמז כי הוא נמצא בתפריט המשנה Send To המסונף לתפריט File.



תרשים 6.5: עורך התפריטים מאפשר ליצור מערכת תפריטים היררכית בצורה מהירה

הערה:

לפעמים קורה ששמו של פריט כלשהו בתפריט מסתיים בשלוש נקודות (...) כמו באפשרויות Open ו-Print בדוגמה לעיל. סימון זה מרמז שקיים מידע נוסף, כגון שם קובץ ומיקום, הדרוש כדי להשלים את הפקודה המבוקשת. לאחר בחירת הפריט, התוכנית תדרוש את המידע הנוסף באמצעות תיבת דו-שיח שתופיע על המסך.

כעת, לאחר שחזרת לסביבת העיצוב של Visual Basic, תוכל לבחור את תפריט **File** בטופס **Form1** כדי לראות כיצד הוא נראה. ניתן לאחד את פקודות הטיפול בקבצים **New**, **Open** ו-**Save**, כקבוצה אחת ואת הפקודות **Print** ו-**Send To**, כקבוצה אחרת. תוכל לבצע זאת, על ידי החלפת מקומות בין האפשרות **Print** והאפשרות **Save**, כך שכל הפקודות הקשורות בטיפול בקבצים תהיינה ביחד. תוכל להבליט את האיחוד וההפרדה באמצעות הוספת **מפרידים** לתפריט.

איחוד פריטים בתפריט (הוספת מפרידים)

בנוסף לשימוש ברמות שונות לצורך ארגון הפריטים בתפריט, יש לפעמים צורך בהפרדת פריטים הנמצאים באותה הרמה. הוספת **מפרידים** בתפריט שוברת את רצף הפריטים ויוצרת הפרדה ברורה בין קבוצות פריטים בעלי מכנה משותף, ללא צורך ביצירת רמות תפריט נוספות. הקש סימן מינוס (-) במאפיין Caption של הפריט כדי להפוך אותו למפריד ברוחב המלא של התפריט שבו הוא מופיע. זכור שאנו צריכים לתת לכל מפריד כזה ערך זיהוי מיוחד במאפיין Name, בדיוק כמו בכל הפריטים האחרים. אנחנו משתמשים בדרך כלל במספרים עולים לזיהוי מפרידים בתפריטים - mnuHyphen1, mnuHyphen2 וכך הלאה.

אזהרה:

אין אפשרות להשתמש במפריד ברמת התפריטים העליונה (שורת התפריטים). ניתן לשלב אותם בתפריטי משנה בלבד. אם תנסה להציב מפריד בשורת התפריטים, תופיע על המסך הודעת שגיאה בעת שתנסה לשמור את פעולתך.

שינוי התפריט

לעיתים קרובות תרצה לבצע שינויים במבנה התפריט שיצרת. תוכל בנקל לבצע משימה זאת באמצעות עורך התפריטים. טבלה 6.1 מפרטת חלק מפעולות העריכה השכיחות ואת אופן ביצוען.

טבלה 6.1: שינוי התפריט

פונקציית העריכה	**דרך הביצוע**
הזזת פריט	בחר את הפריט ולחץ על אחד מלחצני החיצים כדי להזיז אותו למעלה או למטה. הרמה שבה נמצא הפריט אינה משתנה כאשר מזיזים אותו בתפריט.
הוספת פריט בתוך הרשימה	בחר את הפריט האמור להופיע מתחת לפריט החדש והקש על Insert. פריט ריק אמור להופיע מעליו. בשלב זה תוכל למלא את שדות Caption ו-Name שלו. הפריט החדש יימצא ברמה שבה נמצא הפריט שמתחתיו.
הסרת פריט	בחר את הפריט והקש על מקש המחיקה (Delete). הפריט יימחק מיידית ללא אזהרה. בעורך התפריטים אין אפשרות לבטל פעולה. כאשר מוחקים פריט אין דרך להחזיר אותו. אולם אם כתבת קוד תוכנית המשויך לפריט בתפריט, הקוד יישאר וניתן עדיין לראות אותו בחלון הקוד. **הערה**: קוד שנכתב עבור כל פקד לא יימחק עם מחיקת הפקד אלא יישאר בצורתו המלאה ויימחק רק על ידי מחיקת הקוד בחלון הקוד.

כעת תוכל להשתמש בטכניקות העריכה הללו כדי לאחד פריטים ולהוסיף מפרידים אשר נזכרו בחלק הקודם. לשם כך, בצע את המהלכים הבאים:

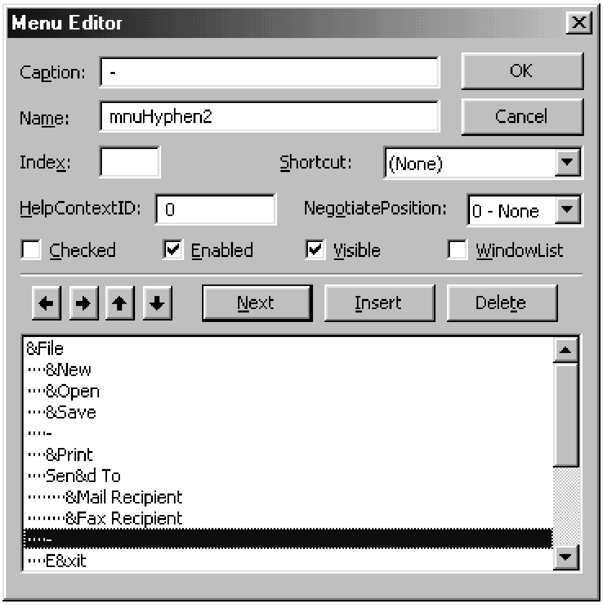
1. פתח את עורך התפריטים לאחר בחירת טופס Form1.
2. לחץ על הפריט **Print** פעם אחת כדי לבחור אותו.
3. לחץ פעמיים על החץ הפונה למטה כדי להזיז את הפריט מעל לפריט **Send To** (לחץ על החץ הפונה למעלה אם יש צורך להזיז פריט מסוים למעלה).
4. כאשר הפריט Print מואר, הקש על מקש **Insert** כדי ליצור פריט ריק מעל **Print**.
5. בחר את מאפיין **Caption** והקלד סימן מינוס (-) בודד.

6. עבור למאפיין **Name** בעזרת TAB והקלד **mnuHyphen1**.

7. בחר את פריט **Exit** והקש על **Insert** כדי ליצור פריט ריק מעל **Exit**.

8. הקלד סימן מינוס (-) במאפיין **Caption** של הפריט החדש ו-**mnuHyphen2** במאפיין השם.

9. לחץ **OK** לסגירת עורך התפריטים.



תרשים 6.6: השימוש ברמות שונות של תפריט מסייע בארגון האפשרויות הקיימות.

מקשי גישה ומקשי קיצור להפעלה מהירה של פונקציות

ניתן להגיע לפריטים רבים הנמצאים בתפריטים של תוכניות Windows על ידי שימוש בצירופי מקשים. תוכל באופן זה לאפשר למשתמש להגיע לפריטים בתוכניותיך בצורה קלה ומהירה. קיימות שתי שיטות חלופיות בהן תוכל להשתמש להגשמת המטרה: **מקשי גישה ומקשי קיצור**.

מקשי גישה

מקש מעין זה מאפשר למשתמש לנווט במערכת התפריטים על ידי שימוש במקלדת במקום בעכבר. במידה ומוגדר עבור הפריט מקש גישה, הוא יסומן באמצעות קו תחתון (לדוגמה, האות F ב-File). נוכל להגדיר מקש גישה על ידי הצבת סימן & לפני האות הרצויה במאפיין **Caption**. בתפריט File לדוגמה, מאפיין Caption הוא &File. נוכל ליצור מקש גישה לכל פריט בתפריט.



טיפ:

המשתמש מצפה שהאות הראשונה בשם הפריט תהווה מקש גישה. לפיכך, מומלץ להשתדל ולהגדיר את מקשי הגישה באופן זה. מצד שני, לא כדאי להשתמש במקש גישה זהה עבור שתי אפשרויות שונות באותו תפריט. לדוגמה, מקש הגישה בתפריט Format ב- Visual Basic הינו האות השנייה במילה. זאת כדי למנוע התנגשות עם תפריט File המופיע גם הוא כאפשרות בשורת התפריטים. תיאורטית, קיימת אפשרות להגדיר מקשי גישה זהים למספר פריטים באותו תפריט. במצב זה, Visual Basic מעבירה את הסמן בין האפשרויות, כלומר עליך לדלג דרך בין הפריטים עבורם מוגדר מקש גישה זהה, עד שתגיע לתפריט המבוקש. כאמור, שיטה זו אינה מומלצת ומפספסת את ייעודה המקורי - להוות קיצור דרך ולחסוך בזמן. בנוסף, משתמשים רבים אינם מודעים ליכולת לדלג בין תפריטים על ידי לחיצה על אותו צירוף המקשים.

כאשר אנו מגדירים מקשי גישה בתפריט, המשתמש יוכל לבחור פריט ברמה הגבוהה (בשורת התפריטים) על ידי החזקת מקש Alt והקשה על מקש הגישה השייך לפריט. פעולה זו גורמת לתפריט המשנה השייך לפריט להיפתח ולהראות את הפריטים השייכים לקבוצה זו. לדוגמה, במקרה שרוצים להגיע לאפשרות **New** הנמצאת בתפריט File, המשתמש יקיש Alt+F ולאחר מכן N.

כדי ליצור מערכת יעילה של מקשי גישה, יש צורך בהגדרת מקש נפרד לכל אחד מהתפריטים הראשיים. לאחר מכן נגדיר מקשים שונים לכל אחד מהפריטים בתפריטי המשנה. תיאורטית, ניתן להגיע רק ל-36 צירופים שונים באותו התפריט (אחד לכל אות באלפבית הלועזי ולכל אחת מעשר הספרות), אולם סביר להניח שייגמר המקום על המסך לפני שתיגמרנה האפשרויות.

מקשי קיצור

בנוסף למקשי הגישה, נוכל להגדיר מקשי קיצור לפונקציות היותר שימושיות בתוכנית. מקשי קיצור מספקים גישה ישירה לפונקציות בעזרת מקש בודד (כגון מקש Delete) או צירופי מקשים (Ctrl+S), ומאפשרים לעקוף לגמרי את מערכת התפריטים. ניתן להשתמש במקשי הקיצור כדי לבצע פעולות בצורה מהירה יותר.

כדי להגדיר מקשי קיצור לפונקציה בתפריט, היכנסו לעורך התפריטים, בחרו את הפריט המבוקש ובחרו את המקש הרצוי מרשימת **Shortcut**. המקש יוצמד לפריט ותיאור מקש הקיצור יופיע בתפריט ליד הפריט.

להדגמת השימוש, שייך את הצירוף Ctrl+P לפריט Print בדוגמה שעשינו בדרך הבאה:

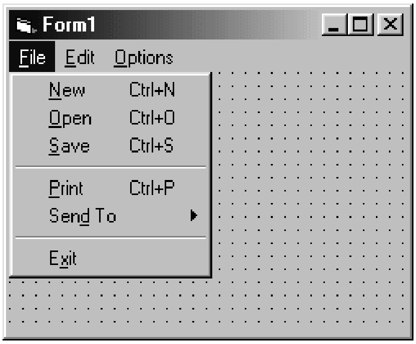
1. ודא שבחרת את **Form1** ופתח את עורך התפריטים.
2. לחץ על הפריט **Print** כדי לבחור אותו.
3. לחץ על החץ הפונה מטה (הנמצא ליד רשימת הקיצורים) ובחר **Ctrl+P** מתוך הרשימה.

4. הגדר את הקיצורים הבאים בעצמך:

הפריט	קיצור דרך
File, New	Ctrl+N
File, Open	Ctrl+O
File, Save	Ctrl+S
Edit, Cut	Ctrl+X
Edit, Copy	Ctrl+C
Edit, Paste	Ctrl+V

5. לחץ **OK** כדי לסגור את עורך התפריטים.

כעת, לאחר שחזרת למצב עיצוב, לחץ על תפריט **File** כדי לראות כיצד צירופי המקשים מוצגים. תרשים 6.7 מדגים את התוצאות.



תרשים 6.7: לחלק מהפריטים שבתפריט File הוקצו מקשי קיצור

ניתן להקצות לכל פריט כל מקש קיצור שהוא, בתנאי שאינו מצוי בשימוש על ידי פריט אחר. עם זאת, מומלץ להשתמש בקיצורים "סטנדרטיים" ושכיחים בהם עושות שימוש תוכניות רבות. בטבלה 6.2 מוצגים חלק ממקשי הקיצור הנפוצים. עמודת ה**תיאור** מסבירה, כדוגמה, מה מקשים אלה מבצעים כאשר משתמשים בסביבת Visual Basic עצמה.



טיפ:

בדומה למקשי הגישה, יש צורך להגדיר מקש קיצור אשר מתאים לאות הראשונה של שם הפריט. לדוגמה, Ctrl+P עבור Print. למשתמש יהיה קל יותר לזכור את הקיצורים בדרך זו. מומלץ להשתמש בקיצורים המתוארים בטבלה 6.2 כדי למנוע בלבול.

טבלה 6.2: שימושים נפוצים בקיצורי דרך לקבלת גישה מהירה לפונקציות בתוכנית

שם הפריט	מקש קיצור	תיאור
Edit, Cut	Ctrl+X	מוחק את הטקסט המסומן ממקומו ומעתיק אותו ללוח.
Edit, Copy	Ctrl+C	מעתיק את הטקסט המסומן ללוח.
Edit, Paste	Ctrl+V	מדביק את תוכן הלוח לטופס הפעיל.
Edit, Undo	Ctrl+Z	מבטל את השינוי האחרון.
Edit, Find	Ctrl+F	פותח את תיבת החיפוש ומאפשר חיפוש מידע כלשהו (כגון טקסט).
File, New	Ctrl+N	יוצר טופס חדש.
File, Open	Ctrl+O	פותח תיבת דו-שיח המאפשרת לפתוח קובץ קיים.
File, Save	Ctrl+S	שומר את הטופס הנוכחי.
File, Print	Ctrl+P	פותח את תיבת דו-שיח המאפשרת לבחור פריטים המיועדים להדפסה.

כתיבת קוד עבור פריטי התפריט

לאחר יצירת מבנה התפריט, יש צורך בכתיבת קוד כדי לגרום לאפשרויות בתפריט לבצע פעולה ממשית. נעשה זאת, כמו בטופס או בפקדים אחרים, על ידי כתיבת קוד בשיגרה המקושרת לאירוע. לכל אחד מן הפריטים בתפריט יש שיגרה אחת בלבד: שגרת Click. שיגרה זו מופעלת כאשר המשתמש לוחץ על הפריט, או כאשר המשתמש מסמן את הפריט ומקיש Enter. לחיצה על הפריט דומה ללחיצה על לחצן פקודה.

הערה:

שגרת Click המקושרת לפריט מופעלת הן כאשר המשתמש מקיש על מקש גישה והן כאשר הוא מקיש על מקש קיצור כדי להפעיל את הפריט.

כדי להוסיף קוד לשגרת Click של הפריט, לחץ קודם כל על הפריט בטופס. פעולה זו גורמת לפתיחת חלון הקוד ומכינה את שגרת האירוע לפריט הנבחר באותה הצורה שבה היתה נפתחת שגרת Click של אובייקט אחר אם היינו בוחרים אותו. לאחר מכן, הקלד את הקוד הגורם לפעולת הפריט.

כעת תוכל לשפר את דוגמת התפריט על ידי תכנות פונקציית File, Exit בתפריט, כמתואר בשלבים הבאים:

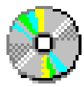
1. בסביבת העיצוב, בחר בתפריט **File** של טופס **Form1** כדי לפתוח אותו ולאחר מכן בחר **Exit**. חלון הקוד ייפתח ויציג את שגרת Click של פריט mnuFileExit.

2. הקלד את הקוד הבא לשיגרה:

```
Dim nTemp As Integer, sTemp As String
sTemp = "Are you sure you want to exit?"
nTemp = MsgBox(sTemp, vbYesNo, "Menu Sample Program")
If nTemp = vbYes Then
      End
End if
```

3. סגור את חלון הקוד.

שמור את תוכנית הבדיקה והרץ אותה. כאשר תבחר באפשרות Exit בתפריט File במערכת התפריטים, תוצג תיבת הודעה אשר תשאל אם אתה בטוח שאתה רוצה לסגור את התוכנית. אם תבחר Yes, התוכנית תיסגר.



בתקליטור:

הקוד שלעיל נמצא בתקליטור בתיקיה Menu Sample Program.

הגדרות נוספות

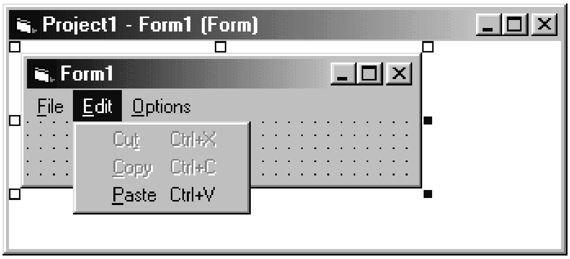
בנוסף למאפיינים הדרושים Caption ו-Name, לכל פריט מספר מאפיינים שניתן להגדיר כדי לשלוט על התנהגות התפריט או כדי לוודא סטטוס של אפשרות מסוימת בתוכנית. המאפיינים השכיחים ביותר בשימוש הינם Visible, Enabled ו-Checked.

המאפיינים Visible ו-Enabled

מאפייני Visible ו-Enabled של הפריט פועלים בדיוק כמו בכל אובייקט אחר. כאשר מאפיין Visible מוגדר True, הפריט מופיע בתפריט. אם המאפיין מוגדר False, הפריט (וכל תפריט משנה המקושר אליו) מוסתר. בוודאי ראית כבר שימוש במאפיינים אלה במעבד תמלילים בזמן שניתן היה לראות את תפריטי File ו-Help בלבד עד שפתחת מסמך חדש. לאחר פתיחת המסמך, התגלו התפריטים האחרים.

שינוי ערך המאפיין Visible מאפשר לשלוט בכמות הפריטים אשר יהיו זמינים למשתמש בכל זמן נתון בתוכנית. בצורה זו של שליטה ניתן למנוע מהמשתמש גישה לפונקציות בתפריט אשר עלולות לגרום לתקלות במידה ותנאים מסוימים לא מתקיימים (לא נרצה לאפשר גישה לפונקציות עריכה כאשר אין טופס פתוח).

מאפיין Enabled משמש בצורה דומה למאפיין Visible. השינוי העיקרי הוא שכאשר ערכו False, הפריט יהיה אפור, גלוי, אך לא זמין. לדוגמה, אין לאפשר Cut ו-Copy אם לא נבחר קטע, אולם אין מניעה להראות שהפונקציות קיימות (ראה תרשים 6.8).



תרשים 6.8: פריטי תפריט חסומים עדיין גלויים לעינו של המשתמש, אולם מוצגים באפור על מנת לציין כי הם מנוטרלים זמנית

אם כי ניתן להגדיר את המאפיינים Visible ו-Enabled במצב עיצוב באמצעות עורך התפריטים, נהוג לרוב לשנות את ערכיהם בזמן ריצה בתגובה לשינויים במצב התוכנית. כמו באובייקטים אחרים, ניתן להגדיר בקוד ערך עבור מאפיין המשויך לפקד תפריט, על ידי ציון המאפיין Name של הפריט, הגדרת שם המאפיין המיועד לשינוי, והערך החדש.

בהמשך לדוגמה המתוארת, ניתן לשפר את מערכת התפריטים הפשוטה שלך. כאשר התוכנית מתחילה לפעול, אין טופס פתוח ולכן לא כדאי לאפשר גישה לאפשרויות **Save**, **Send to** ו-**Print** כמו גם את תפריט **Edit** אשר תרצה להציגו רק במצב בו יש טופס פתוח. בצע את הפעולות הבאות כדי לוודא שמערכת התפריטים תוצג בצורה הנכונה עם הפעלת התוכנית:

1. ודא שטופס **Form1** מסומן. היכנס לעורך התפריטים.
2. לחץ על הפריט **Save** וסמן אותו.
3. הסר את הסימון הנמצא על מאפיין **Enabled**. פעולה זו משנה את ערך מאפיין Enabled ל-False.
4. הגדר ערך **False** גם לפריטים **Send to** ו-**Print**.
5. בחר את תפריט **Edit**.
6. הסר את הסימון הנמצא על מאפיין **Visible**. פעולה זו מגדירה את ערך המאפיין כ-False.
7. סגור את עורך התפריטים.



טיפ:

שים לב שתפריט Edit נעלם גם מסביבת העיצוב. עדיין תוכל להגיע לשגרת Click של התפריט על ידי כניסה לחלון הקוד של הטופס ובחירה בתפריט המתאים מתוך תיבת האובייקטים. תוכל גם להשאיר את מאפיין Visible במצב True ולהגדיר אותו כ-False בשגרת Load של הטופס.

בצע את הפעולות הבאות כדי לאפשר את החלקים המתאימים של תפריט File, וכדי להראות את תפריט Edit בזמן שקיים טופס פתוח:

1. בסביבת העיצוב, לחץ על תפריט File ולחץ על הפריט Open. שגרת **mnuFileOpen** תוצג.
2. בתוכנית אמיתית נציב כאן את הקוד המטפל בפתיחת טופס. תוכל להציב שורת הערה המסבירה את מהות הקטע. הקלד את השורות הבאות לתוך שגרת mnuFileOpen_Click:

```
'Code for opening a file goes here
mnuFileSave.Enabled = True
mnuFileSendTo.Enabled = True
mnuFilePrint.Enabled = True
mnuEdit.Visible = True
```

3. סגור את עורך התפריטים.
4. שמור את התוכנית והפעל אותה. שים לב שתפריט Edit אינו נראה. כמו כן, שים לב שכאשר הנך לוחץ על תפריט File, חלק מהפריטים בתפריט מופיעים באפור.
5. בחר **File, Open** הפעולה תגרום לשגרת Click של פריט Open להתבצע. בחר שוב בתפריט File ושים לב שכל הפריטים בתפריט זה מאופשרים. כמו כן, שים לב שניתן כעת לראות את תפריט Edit.
6. לחץ על לחצן הסיום בסרגל הכלים כדי לסגור את התוכנית.

מאפיין Checked

מאפיין זה קובע אם יוצג סימון (Check Mark) משמאל לפריט בתפריט, כמו שניתן לראות בתרשים 6.9. משתמשים בו כדי להציג מצב של אפשרות מסוימת בתוכנית. לדוגמה, אם משתמש בחר להשתמש באפשרות מסוימת, יופיע לידה סימון (✓). ניתן לבחור שוב את האפשרות בתפריט כדי להסיר את הסימון ושוב כדי להחזירו.



תרשים 6.9: מאפיין Checked קובע אם יוצב סימון לשמאל הפריט

הקלד את הקוד הבא לתוך שגרת Click של הפריט mnuOptionsText בתפריט הדוגמה:

```
mnuOptionsText.Checked = Not mnuOptionsText.Checked
```

מאחר וערך מאפיין Checked יכול להיות True או False בלבד, הקוד שכתבת עושה שימוש יפה בלוגיקת חשיבה בוליאנית בעזרת הפעולה Not. הפעולה מייצגת את ההפך של ערך בולאני כלשהו. לדוגמה: הפונקציה Not mnuOptionsText.Checked מחזירה את היפוכו של הערך הנוכחי של מאפיין Checked בפריט mnuOptions. התהליך ידוע כ**היפוך** (Toggling) ערך בולאני. בכל נקודה בתוכנית, מצב מאפיין Checked קובע אם המשתמש רוצה שהטופס יהיה מורכב מטקסט בלבד או לא.

כמובן שניתן להגדיר את מאפיין Checked כבר בסביבת העיצוב אם נרצה שהפריט המדובר יהיה מסומן כאשר התוכנית מתחילה לפעול.

אזהרה:

אין אפשרות להגדיר את מאפיין Checked של פריט הנמצא ברמה העליונה ביותר (על שורת התפריטים) כ-True. אם תעשה זאת תופיע הודעת שגיאה.

מאפיינים אחרים

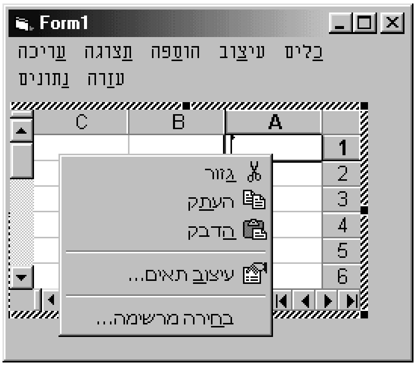
בוודאי שמת לב לשתי אפשרויות נוספות בעורך התפריטים. אפשרויות המגדירות את ערך מאפייני NegotiatePosition ו-WindowList של הפריט. NegotiatePosition קובע אם והיכן יוצג הפריט בתוכנית, בעת הצגת תפריט המקושר לאובייקט המוטמע (Embedded) בטופס (דוגמה, אם קיים מנגנון Word פעיל המשמש לעריכת טופס). כאשר הערך של מאפיין NegotiatePosition הוא 0, התפריט לא יוצג בזמן שהאובייקט פעיל. כאשר הערך שונה מאפס, הפריט יוצג לשמאל, באמצע או לימין התפריט המקושר לאובייקט (על ידי קביעת הערכים 1,2,3 בהתאם). מאפיין WindowList קובע האם לתפריט הנוכחי תצורף רשימה דינאמית של טפסי משנה פעילים ביישומי MDI. כאשר ערך מאפיין זה הוא True, הפריטים יתווספו אוטומטית לתפריט בעת פתיחת **טפסי בן** (Child Forms) ויוסרו כאשר הטפסים הללו ייסגרו. תוכל ללמוד יותר על נושא זה בפרק 17 **יישומים בעלי ממשק מרובה מסמכים**.

יצירת תפריטים מקוצרים

עד כאן למדת על שורת התפריטים המופיעה בראש החלון. Visual Basic תומכת גם ב**תפריטים מקוצרים** הנקראים גם **תפריטים מוקפצים** (Pop-Up Menus). הכוונה היא לתפריטים קטנים המוצגים במקום כלשהו בטופס, כתגובה לאירוע בתוכנית.

בדרך כלל משתמשים בתפריטים מקוצרים כדי לטפל בפעולות או אפשרויות השייכות לאזור מסוים בטופס (ראה תרשים 6.10) - לדוגמה, תפריט מקוצר המופיע בשדה טקסט מאפשר לשנות את הגופן או את מאפייני הגופן בשדה. תוכל למצוא תפריטים כאלה ברבות מתוכניות Windows מהדור החדש, כולל Visual Basic עצמה.

בעת הפעלת תפריט מקוצר, בדרך כלל על ידי לחיצה ימנית על אובייקט כלשהו, הוא מופיע על המסך ליד מצביע העכבר. במצב זה, המשתמש בוחר באפשרות מתוך התפריט. לאחר שנבחרה אפשרות, התפריט נעלם מהמסך.



תרשים 6.10: התפריט המקוצר, אשר מקושר לרשת, מהווה דרך נוחה להפעלת פונקציות בתוכניות המיוחדות לרשת

יצירת התפריט

תוכל ליצור תפריט מקוצר בדומה לאופן בו יצרת את התפריט הראשי בתוכנית - בעזרת עורך התפריטים. אולם, עליך לבצע צעד נוסף. התפריט המקוצר אמור להיות חבוי כך שלא יוצג על גבי שורת התפריטים. כדי להחביא את התפריט, עליך לקבוע את ערך המאפיין Visible של התפריט, ברמה הגבוהה ביותר, כ-False.

הערה:

ככלל, תרצה להחביא את פריט התפריט אשר משמש כתפריט מקוצר. אולם, באפשרותך להשתמש בכל אחד מפריטי הרמה העליונה גם כתפריט מקוצר. כלומר, תפריט מסוים יכול להיות גם תפריט מקוצר וגם חלק מהתפריט הראשי של הטופס.

כדי להדגים את התפיסה של תפריט מקוצר, הוסף תפריט פשוט לתפריט הדוגמה:

1. בעזרת עורך התפריטים, הוסף פריט חדש ברמה העליונה בסוף רשימת הפריטים. קבע את ערך מאפיין **Caption** כ-**Format** ואת שמו כ-**mnuFormat**.
2. הגדר את מאפיין **Visible** של פריט **mnuFormat** כ-**False**.
3. צור את הפריטים הבאים תחת תפריט Format:

Name	Caption
mnuFormatBold	&Bold
mnuFormatItalic	&Italic
mnuFormatUnderline	&Underline

4. סגור את עורך התפריטים.

שים לב שהתפריט Format אינו מופיע בשורת התפריטים. אולם הוא קיים וניתן לערוך אותו בעורך התפריטים.

גם הטכניקה להוספת קוד לשגרת Click של פריטים בתפריט מקוצר שונה מעט. אין אפשרות לבחור בפריט כדי לקרוא לחלון הקוד מפני שהתפריט אינו נראה לעין. אך תוכל לפתוח את חלון הקוד על ידי לחיצה על הלחצן View Code בחלון הפרויקט או לחיצה כפולה על הטופס. אחר תוכל לבחור את הפריט המבוקש בתיבת האובייקטים הנמצאת משמאל למעלה. בדרך זו, תוכל לכתוב קוד עבור הפריטים החבויים.

הפעלת תפריט מקוצר

כדי להציג תפריט מקוצר על המסך, יש צורך להשתמש בשיטת PopUpMenu. תוכל לעשות זאת על ידי קביעת שם הטופס שבו התפריט יוצג, שיטת PopUpMenu, ושם התפריט המבוקש. למרות שניתן להשתמש בשיטה זו בכל מקום בתוכנית, משתמשים בתפריטים מקוצרים בעיקר כתגובה ללחיצות על הלחצן הימני של העכבר.

כדי להדגים את השימוש בשיטה זו, צור תפריט מקוצר לדוגמה אשר מופעל כאשר המשתמש לוחץ לחיצה ימנית על הטופס. אומנם בתוכנית מלאה המטפלת בטקסט, סביר כי המשתמש ילחץ לחיצה ימנית על טקסט נבחר, אך הצעדים הבאים מדגימים את הרעיון בכללותו:

1. פתח את חלון הקוד של הטופס על ידי לחיצה כפולה על הטופס.
2. פתח את תיבת האובייקטים בחלון הקוד ובחר את הפריט **mnuFormatBold**.
3. בשגרת **Click** של פריט **mnuFormatBold**, הקלד את שורת הקוד הבאה, אשר תגרום לתיבת הודעה להופיע ולדווח שהשיגרה אמנם הופעלה:

```
MsgBox "You just bolded some text!"
```

4. פתח את תיבת האובייקטים בחלון הקוד ובחר **Form**.
5. פתח את תיבת השגרות בחלון הקוד ובחר את שגרת **MouseUp**.
6. הקלד את הקוד הבא לתוך שגרת **MouseUp** של הטופס:

```
If Button = vbRightButton Then
      Form1.PopupMenu mnuFormat
End If
```

7. שמור את התוכנית והפעל אותה. לחץ לחיצה ימנית על הטופס כדי לפתוח את תפריט **Format** ובחר באפשרות **Bold**.

בקטע הקוד שהקלדת זה עתה, השתמשת בשגרת MouseUp לביצוע פעולה כאשר נלחץ לחצן בעכבר. השיגרה העבירה משתנה בשם Button המודיע איזה מלחצני העכבר נלחץ. מפני שרצינו שהתפריט יופיע רק במידה ונלחץ הלחצן הימני של העכבר, בדקנו את ערך משתנה Button. אם נמצא כי הוא שווה ערך לקבוע vbRightButton (קבוע פנימי עבור ערך הלחצן הימני של העכבר), הוצג התפריט המקוצר.

הערה:

תוכל ליצור מספר תפריטים מקוצרים ולהציג כל אחד מהם כתגובה ללחיצה על לחצנים שונים בעכבר, או באזורים שונים על גבי המסך. ערכי X ו-Y המועברים לשגרת MouseUp מדווחים על מיקום מצביע העכבר בזמן התרחשות האירוע.

שימוש בסרגלי כלים ב- Visual Basic

בוודאי שמת לב לכך שבתוכניות רבות המבוססות על Windows ישנם סרגל כלים אחד או שניים בנוסף למערכת התפריטים. סרגלי כלים אלה מספקים למשתמש דרך גישה מהירה לרוב הפונקציות השכיחות בתוכנית. מספר תוכניות אף משתמשות בסרגלי כלים לצורך ביצוע פעולות מיוחדות, כגון סרגל ציור הקיים ב- Microsoft Word. כאשר סרגלי הכלים הפכו נפוצים, המשתמשים התרגלו לצפות להם בכל תוכנית.

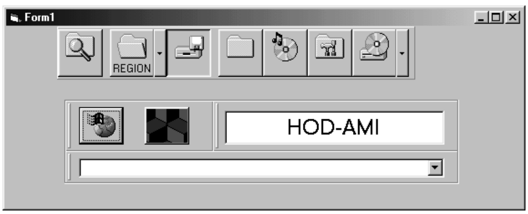
תוכל ליצור סרגל כלים בצורה מהירה ב- Visual Basic. למעשה, Visual Basic 6 מאפשרת שני סוגים של סרגלי כלים: הסוג הרגיל **והסוג המשופר**, **CoolBar**.

עקרונות סרגל הכלים

למרות ששני סוגים אלה של סרגלי כלים (הנראים בתרשים 6.11) מופיעים למשתמש בצורה שונה, שניהם משמשים לאותה מטרה: לספק מערכת של לחצנים מאוירים, המאפשרים גישה לפונקציות שכיחות בתוכנית.

פקד **ToolBar** מאפשר ליצור שישה סוגים שונים של לחצנים:

- לחצנים מסוג Pushbuttons הפועלים כמו לחצני פקודה.
- לחצני סימון (Checkbuttons) הפועלים במצבי הפעל-הפסק, כמו תיבת סימון.
- קבוצות לחצנים (Button groups) הפועלות כמו לחצני אפשרויות (Option buttons).
- לחצני הפרדה (Seperator buttons) היוצרים רווחים בסרגל הכלים.
- לחצני מרווח (Place holder buttons) היוצרים שטח ריק אשר בו תוכל להציב פקדים אחרים, כגון (Combo box), על גבי סרגל הכלים.
- לחצני תפריטים נפתחים (Drop-down buttons) מהם נפתחים תפריטים המציגים אפשרויות שונות בפני המשתמש.

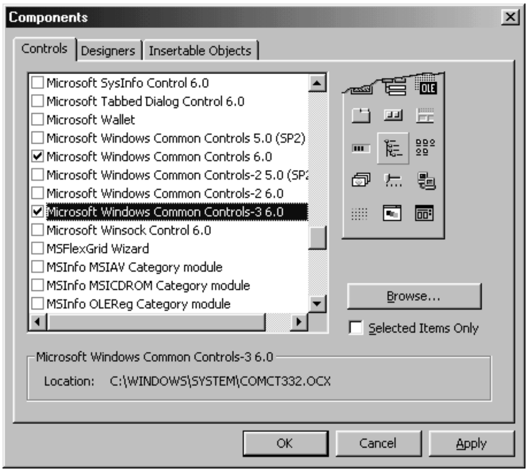


תרשים 6.11: תפריט מסוג Coolbar הוצג לראשונה בתוכנת Internet Explorer

כדי ליצור סרגל כלים, יש צורך להשתמש בפקדים שונים: אחד או יותר מפקדי ToolBar ופקד ImageList. פקד ImageList מכיל אוסף ציורים הניתנים לשימוש על ידי פקדים אחרים. במקרה זה, פקד ToolBar מציג ציורים מתוך פקד ImageList על גבי הלחצנים. פקד ImageList הינו חלק מתוך קבוצת Microsoft Windows Common Controls 6.0. מידע נוסף על הקבוצה ניתן לקבל בפרק 12, **הפקדים המשותפים של Microsoft**.

ראה: פקד ImageList: פקד משותף בסיסי, פרק 12.

כדי ליצור פקד מסוג ToolBar על הטופס, יש צורך בהוספתו קודם לכן לארגז הכלים. לחץ לחיצה ימנית באזור ריק של ארגז הכלים ובחר **Components** מתוך התפריט שיופיע. כאשר תיבת הדו-שיח של הרכיבים מופיעה, בחר את הקבוצה הראשונה של הפקדים השכיחים - **Microsoft Windows Common Controls 6.0** - כמוצג בתרשים 6.12. קבוצת פקדים זו מכילה את פקדי סרגל הכלים הרגילים ואת פקד ImageList. CoolBar נמצא בקבוצה השלישית (Microsoft Windows Common Controls - 3 6.0). בחר בקבוצה זו כדי להוסיף את פקד CoolBar לארגז הכלים.



תרשים 6.12: פקד CoolBar הוא קבוצה חדשה - Microsoft Windows Common Controls - 3 6.0.

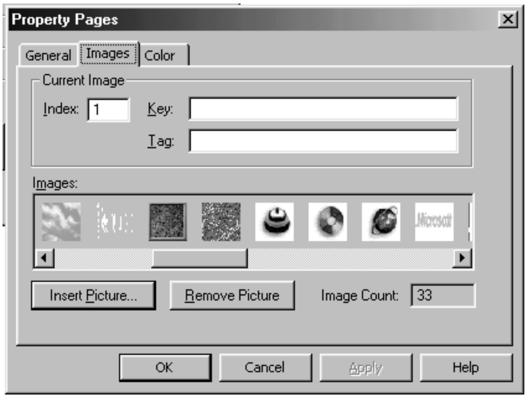
בחלקים הבאים תגלה כיצד לכלול את שני סוגי סרגלי הכלים בתוכניות שלך.

קביעת התמונות בסרגלי הכלים

השלב הראשון ביצירת סרגל כלים כלשהו, כולל הוספת התמונות, אשר תוצגנה על לחצני הסרגל, לפקד ImageList. בצע את הפעולות הבאות:

1. שרטט פקד מסוג ImageList על הטופס ותן לו שם ייחודי. גודל הפקד ImageList נקבע על ידי Visual Basic, זאת משום שהוא אינו מופיע בעת פעולת התוכנית. לכן הוא נראה כתווית קטנה בזמן העיצוב, ללא קשר לצורה שבה הוא שורטט.

2. להוספת תמונות מסוג Bitmap לפקד, פתח את תיבת המאפיינים שלו על ידי לחיצה על הלחצן **בנייה** (...) המופיע בחלון המאפיינים בשורת המאפיין **Custom**, או על ידי לחיצה ימנית על פקד **ImageList** ובחירת **Properties**. מתיבת דו-שיח זו, הנראית בתרשים 6.13, תוכל להוסיף תמונות מתוך קבצים גרפיים (מסוג Icons ו-Bitmaps) הנמצאים במחשב.



תרשים 6.13: בזמן העיצוב, תוכל להוסיף ציורים לפקד ImageList על ידי שימוש בכרטיסיית הציורים הנמצאת בתיבת הדו-שיח של המאפיינים

3. כדי להוסיף תמונה לפקד, לחץ על הלחצן **Insert Picture**. פעולה זו תציג בפניך את תיבת הדו-שיח **Select Picture** (בחירת תמונה), אשר ממנה תוכל לבחור תמונת Bitmap (מפת סיביות) או Icon (סמל), על פי רצונך. לאחר בחירת התמונה הרצויה, היא תתווסף לפקד ותוצג באזור התמונות.

4. לאחר הוספת כל התמונות הרצויות, לחץ על לחצן **OK** וסגור את תיבת הדו-שיח.

הערה:

ב- Visual Basic קיימים ציורים רבים המוטמעים בתוכנה. במידה ובחרת להתקין את הציורים הללו, הם יאוחסנו בתיקיה הקרויה **Graphics** (מיקום התיקיה תלוי באפשרויות ההתקנה שבחרת וכן האם התקנת את Visual Basic לבדה או כחלק מחבילת Visual Studio).

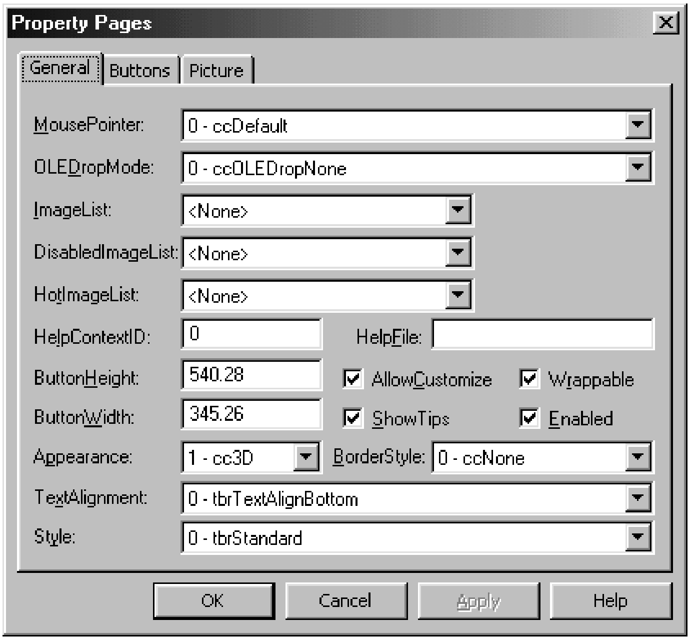
לאחר ביצוע הפעולות הנזכרות לעיל, פקד ImageList שיצרת מוכן לספק ציורים לסרגל הכלים שלך.

יצירת סרגל כלים רגיל

כדי ליצור פקד ToolBar רגיל, יש צורך להציב אותו קודם לכן על גבי טופס. בצע את הפעולות הבאות כדי לעשות זאת:

1. שרטט את פקד Toolbar בטופס. Visual Basic מציבה את הפקד בראש הטופס. אורך הפקד מתאים לאורך הטופס.
2. יישר את סרגל הכלים. ניתן ליישר את הסרגל לכל צד או ליצור **סרגל צף**:

- כדי להציב את הסרגל בראש הטופס, השאר את ערך מאפיין **Align** שווה ל- **1-vbAlignTop**.
- כדי להציב את הסרגל בתחתית הטופס, קבע את ערך מאפיין **Align** כ- **2-vbAlignBottom**.
- כדי ליישר את הסרגל לימין או שמאל הטופס, קבע את ערך מאפיין **Align** כ- **3-vbAlignLeft** או **4-vbAlignRight** בהתאמה. במקרים אלה מומלץ לקבוע את ערך **Width** בצורה ידנית כדי למנוע מהסרגל למלא את שטח הטופס לגמרי.



תרשים 6.14: תוכל להקצות פקד **ImageList** לסרגל הכלים שלך בכרטיסיה General שבגיליון המאפיינים של הסרגל

- כדי ליצור סרגל צף, הגדר את ערך מאפיין **Align** כ- **0-vbAlignNone**. תוכל לקבוע את מיקומו וגודלו על ידי קביעת ערך מאפייני **Width**, **Top**, **Left** ו-**Height**.

3. הצג את מאפייני הפקד על ידי לחיצה ימנית עליו ובחירה באפשרות **Properties**. תרשים 6.14 מציג את הכרטיסיה הראשונה בחלון מאפייני פקד ToolBar.

4. קבע את מאפיין **ImageList** של הפקד כשם פקד **ImageList** שיצרת קודם כדי לספק סמלים לסרגל הכלים. לחיצה על החץ הנמצא מימין למאפיין **ImageList** פותחת רשימה של כל פקדי **ImageList** הקיימים בטופס הנוכחי.

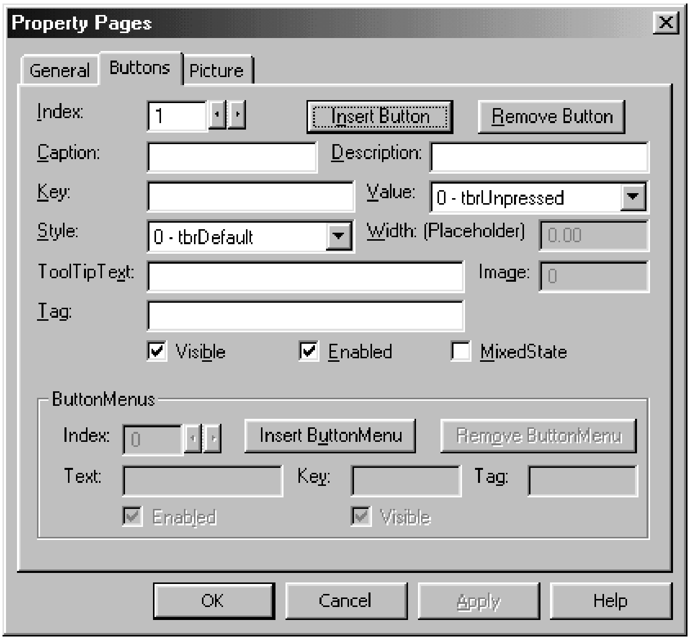
מספר מאפיינים בכרטיסיה General שבגיליון המאפיינים שולטים במראה הסרגל ובצורת התנהגותו. מאפיינים אלה מתוארים בטבלה 6.3.

טבלה 6.3: מאפייני פקד ToolBar השולטים במראהו ובצורת התנהגותו

שם המאפיין	תיאור
BorderStyle	קובע האם תיראה מסגרת מסביב לסרגל הכלים או שלא תיראה מסגרת כלל.
ButtonHeight	מגדיר את גובה הלחצנים בסרגל (ביחידות Twip).
ButtonWidth	קובע את רוחב הלחצנים בסרגל (ביחידות Twip).
AllowCustomize	קובע האם תהיה למשתמש אפשרות לשנות את סרגל הכלים על ידי הוספה, הסרה או הזזת לחצנים.
ShowTips	קובע האם יוצגו תיאורי כלים בשעה שסמן העכבר מוצב מעל אחד הלחצנים.
Wrappable	קובע האם סרגל הכלים יכול לגלוש לשורה הבאה במידה ויש יותר מדי לחצנים עבור שורת לחצנים אחת.
HotImageList	מגדיר פקד ImageList שונה אשר ייספק סמלים עבור לחצנים מסוג DropDown ו-CheckStyle כאשר הם מופעלים. ציור חלופי יוצג גם כאשר מצביע העכבר נמצא מעל כל לחצן אם ערך מאפיין Style הינו 1-tbrFlat. כדי להשתמש במאפיין, ודא שאינדקס הציור המבוקש בפקד ImageList המוקצה, שווה לאינדקס ברשימת הציורים הראשית.
DisabledImageList	מגדיר פקד ImageList שונה אשר ייספק ציורים ללחצני סרגל הכלים שמאפיין Enabled שלהם מוגדר כ-False. כדי להשתמש במאפיין זה, ודא שלציור המבוקש בפקד ImageList יש אותו אינדקס שיש לציור ברשימה הראשית.
Style	קובע אם יתווסף אפקט תלת-מימדי לסרגל הכלים או אם הוא יופיע בצורה שטוחה.

יצירת הלחצנים עבור סרגל הכלים

השלב הבא ביצירת סרגל הכלים יהיה ליצור את הלחצנים אשר יוצבו בסרגל. למשימה זו, עבור לכרטיסיית הלחצנים בגיליון המאפיינים (ראה תרשים 6.15).



תרשים 6.15: תוכל להקצות ציורים, מזהים, ואפילו תפריטים ללחצני סרגל הכלים באזור הלחצנים

יצירת לחצן רגיל לסרגל הכלים

כדי להוסיף לחצן לסרגל, לחץ על לחצן **Insert Button**. במידה וכבר קיימים לחצנים אחרים בסרגל, ייווצר לחצן חדש אחרי הלחצן הנוכחי. עבור כל לחצן שתוסיף יש להגדיר מספר מאפיינים: Style ,Key ו-Image.

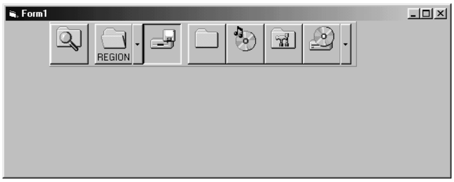
מאפיין **Key** קובע שם אשר מזהה את הלחצן מצד התוכנה. תוכל לראות כיצד משתמשים במאפיין זה בסעיף **כתיבת קוד עבור לחצנים** אליו תגיע עד מהרה. מאפיין Key חייב להיות ייחודי לכל אחד מהלחצנים. כמו כן, יש להקצות שם אשר הינו בעל משמעות עבורך. בצורה זו, תוכל להיזכר במשמעות הלחצן בצורה נוחה יותר כאשר תכתוב את הקוד.

מאפיין **Image** מגדיר את האינדקס עבור התמונה שתופיע על גבי הלחצן. האינדקס מתייחס לאינדקס של התמונה בפקד ImageList. תוכל להגדיר ערך אפס למאפיין אם אינך מעוניין שתופיע תמונה על גבי הלחצן. תיבה זו תהיה אפורה עד אשר יוגדר Imagelist עבור סרגל הכלים.

מאפיין **Style** קובע את סוג הלחצן שברצונך ליצור. טבלה 6.4 מסכמת את הערכים האפשריים למאפיין Style. כל אחד מסגנונות אלה מוצג בתרשים 6.16.

טבלה 6.4: ערכי מאפיין Style הקובעים את צורת ההתנהגות של לחצני סרגל הכלים.

ערך המאפיין	קבוע VB	דוגמה
0 - tbrDefault	הלחצן הינו לחצן רגיל.	לחצן Save Project ב- Visual Basic
1 - tbrCheck	הלחצן מציין שאפשרות מסוימת נמצאת במצב פעיל או לא פעיל לפי מצבו.	לחצן Bold ב-Word
2 - tbrButtonGroup	הלחצן הוא חלק מקבוצה. רק אחד מתוך הקבוצה יכול להיות לחוץ בזמן נתון.	לחצני יישור ב-Word
3 - tbrSeperator	הלחצן משמש כרווח בין לחצנים אחרים. רוחבו שמונה נקודות מסך (Pixels).	אין.
4 - tbrPlaceHolder	משמש לשמירת מקום בסרגל הכלים לפקדים אחרים כגון תיבה משולבת.	תיבת הגופנים ב-Word
5 - tbrDropDown	בשימוש עם ButtonMenu כדי ליצור תיבת רשימה נפתחת בסרגל הכלים	לחצן Add Form ב- Visual Basic



תרשים 6.16: בתרשים זה מוצגות דוגמאות שונות לסגנונות לחצנים בסרגל הכלים. שים לב לכך שלחצני ההפרדה והמרווחים אינם לחצנים ממשיים.

בנוסף לשלושת המאפיינים העיקריים אשר דנו בהם, תוכל לקבוע מספר מאפיינים נוספים לכל לחצן בסרגל הכלים (ראה טבלה 6.5).

טבלה 6.5: מאפיינים המקנים שליטה נוספת בלחצנים (אין חובה לקבוע את ערכם).

מאפיין	תיאור
Caption	הטקסט הנמצא במאפיין זה יוצג מתחת לתמונת הלחצן.
Description	טקסט המתאר את הלחצן בהפעלת תיבת הדו-שיח Customize.
ToolTipText	טקסט זה יופיע כאשר מצביע העכבר נמצא על הלחצן. טקסט זה יופיע רק בתנאי שערך מאפיין ToolTips של סרגל הכלים הינו True.
Value	המאפיין קובע/מחזיר מצב עדכני של הלחצן. ערך 0 מציין שהלחצן אינו לחוץ. ערך 1 מציין שהלחצן לחוץ. בדרך כלל משתמשים בלחצן כדי לקבוע מצב לחצן מסוג CheckStyle או לחצן בתוך קבוצה.

בתום הגדרת הלחצנים על גבי סרגל הכלים, תוכל לצאת מתיבת הדו-שיח **Property Pages** על ידי לחיצה על לחצן **OK**.

טיפ:

במידה והציורים הנמצאים על גבי הלחצנים אינם נותנים תיאור מספק לגבי מהות הלחצן, מומלץ להשתמש ב**תיאורי כלים** (Tooltips). תיאור כלי היא תיבת טקסט קטנה המכילה תיאור קצר של הלחצן. התיבה מופיעה על המסך כאשר המשתמש משהה לרגע את מצביע העכבר מעל ללחצן. בהתאם למאפייני התוכנית שלך, תוכל להשתמש בטכניקת ToolTip במקום בתיאור על גבי הלחצן עצמו, כדי לחסוך מקום במסך. ללא טקסט כלשהו המשתמש לא יוכל לדעת אילו שימושים ניתן לעשות בלחצן וייאלץ ללחוץ עליו כדי לברר מה תפקידו. ניתן להוסיף תיאורי כלים אשר ייתנו לתוכנית מראה מקצועי.

הוספת לחצן מסוג תיבת רשימה נפתחת לסרגל הכלים

פריט חדש ב- Visual Basic 6 הוא לחצן בסגנון tbrDropDown. לחצן זה משלב לחצן רגיל עם תפריט רשימה נפתחת אשר יוצר אפקט הדומה לתיבה משולבת. הדוגמה הבאה מראה כיצד ליצור לחצן כזה ולתכנת את האירועים הקשורים בו.

יצירת סרגל כלים לדוגמה

כדי להדגים את השימוש בסרגלי כלים, חלק זה ידריך אותך ביצירת תוכנית דוגמה המכילה לחצן רגיל ולחצן מסוג Drop Down יחד עם לחצן רווח המספק מרווח ביניהם. בצע את הפעולות הבאות:

1. צור פרויקט רגיל מסוג **EXE**. הוסף פקד **ToolBar** ופקד **ImageList** לטופס כמתואר בסעיפים הקודמים.
2. קבע את מאפיין **Name** של סרגל הכלים כ-**tbrDemo**. השאר את מאפיין **Name** של פקד **ImageList** כשם ברירת המחדל, **ImageList1**.
3. בדפי המאפיינים של **ImageList1** בכרטיסיה **Images**, הוסף שני סמלים מתאימים לפקד **ImageList** (לחץ על לחצן **Insert Picture** ועיין בקבצי התמונות). סמלים אלה יהיו, בסופו של דבר, התמונות אשר יוצגו על גבי שני הלחצנים בסרגל הכלים.
4. לחץ לחיצה ימנית על פקד סרגל הכלים ובחר באפשרות **Properties**. בדפי המאפיינים של סרגל הכלים, קבע את מאפיין **ImageList** של סרגל הכלים כ-**ImageList1** (שם פקד ImageList שלך).
5. בכרטיסיית הלחצנים, לחץ על לחצן **Insert**. מאפיין **Style** אמור להיות במצב ברירת המחדל שלו, **0 - tbrNormal**. קבע את ערך מאפיין **Key** כ-**NewFile**.
6. קבע את ערך **Image** של הלחצן כך שיהיה האינדקס של הסמל המתאים בפקד ImageList (אשר אמור להיות 1 במידה וביצעת נכון את הצעדים הקודמים).
7. לחץ שוב על **Insert** כדי להוסיף לחצן. קבע את **Style** כ- **3 - tbrSeperator**. לחצן ההפרדה ייראה כרווח ריק בין הלחצנים הראשון והשלישי.

8. לחץ שוב על לחצן **Insert** כדי להוסיף לחצן שלישי. קבע את מאפיין **Style** שלו כ- **tbrDropDown - 5**. קבע את ערך **Key** שלו כ-**report** ואת ערך **Image** כאינדקס של הסמל השני בפקד **ImageList** (2). קבע את מאפיין **Caption** כ-**REGION**.

9. כדי להוסיף פריטים נוספים לתפריט המקושר ללחצן זה, לחץ על לחצן **Insert ButtonMenu** (הלחצן הנמצא במסגרת בחלק התחתון של החלון). הקלד **East** במאפיין **Text** של הפריט.

10. חזור על סעיף 9 כדי להוסיף את חלקי **West**, **South** ו-**Central**.

11. לחץ על לחצן **OK** כדי לסגור את דפי המאפיינים. לחצן DropDown אמור להופיע על גבי סרגל הכלים.

כתיבת קוד ללחצנים

ראית כיצד להגדיר את סרגל הכלים, אולם עד שתוסיף קוד לשגרות אירועי סרגל הכלים, התוכנית לא תבצע שום פעולה. ללחצנים בסרגל הכלים אין אירועים משלהם. במקום זאת, יש לכתוב קוד פרטי עבור אירועי **ButtonClick** ו-**ButtonMenuClick** של סרגל הכלים עצמו. אירועים אלה מעבירים את אינדקס **Button** או **ButtonMenu** כמשתנה לשגרת האירוע. בקוד שלך תשתמש בערך מאפיין **Key** של אובייקט הלחצן כדי לקבוע איזה לחצן נלחץ.

על ידי השלמת הפעולות בסעיפים הקודמים, יצרת סרגל כלים עם לחצנים יפים שאפשר ללחוץ עליהם, אבל הם אינם עושים מאומה. יש צורך להוסיף קוד כדי להפוך אותם לשימושיים. כתיבת שגרת אירוע עבור לחצן בסרגל הכלים נעשית בצורה קצת שונה מאשר כתיבת שיגרה ללחצן רגיל. במקום להשתמש בשגרת אירוע נפרדת עבור כל לחצן ותפריט, הקוד שלך חייב לבדוק את ערכי מאפייני האובייקטים כדי לקבוע איזה לחצן או תפריט נבחרו.

הקלד את הקוד המוצג בתוכנית 6.1 לתוך חלון הקוד של הטופס.

תוכנית דוגמה 6.1: TBRDEMO.VBP - שימוש בקוד לצורך הפעלת סרגל כלים

```
Option Explicit
Dim sCurrentRegion As String

Private Sub tbrDemo_ButtonClick(ByVal Button As ComctlLib.Button)
    Select Case Ucase(Button.Key)
        Case "NEWFILE"
            'Code to create a new file goes here
            MsgBox "Creating a new file..."
        Case "REPORT"
            'Create a report with the current region
            Call MakeReport
    End Select
End Sub
```

```
Private Sub tbrDemo_ButtonMenuClick(ByVal ButtonMenu As ComctlLib.ButtonMenu)
    'Create a report using the newly selected region
    sCurrentRegion = UCase(ButtonMenu.Text)
    ButtonMenu.Parent.Caption = sCurrentRegion
    Call MakeReport

End Sub

Sub MakeReport()
Dim sTemp As String
    If Trim(sCurrentRegion) = "" Then
        sCurrentRegion = "WEST"
    End If
    sTemp = "Creating report for " & sCurrentRegion
    sTemp = sTemp & " Region..."
    MsgBox sTemp
    'Actual report - creating lines go here...

End Sub
```

לאחר שהקלדת את הקוד, הפעל את התוכנית ובדוק את התנהגות לחצני סרגל הכלים.

השגרות הנקראות הן למעשה שגרת Click עבור פריטי התפריט. בעזרת השגרות אשר נקראו, תוכל לכתוב קוד אשר יבצע פעולה פעם אחת ואחר לקרוא לו מהתפריט או מסרגל הכלים. דרך זו מקלה על ניהול הקוד מפני שיש צורך לבצע שינויים או תיקונים בחלק אחד בלבד. בנוסף, שים לב לפקודת Select Case אשר מופיעה כאן לראשונה ומהווה תחליף נוח למערכת פקודות If. תוכל ללמוד על כך יותר בפרקים הבאים.

ראה: **שימוש בפקודה Select Case** בפרק 10, **שליטה במהלך התוכנית**.

יצירת סרגל כלים המכיל קוד

למרות העובדה שהינך יכול ליצור סרגל כלים בסביבת העיצוב בלבד, תוכל בנוסף לכך ליצור ולשנות סרגל כלים בזמן פעולת התוכנית בעזרת קוד. כדי להשתמש בסרגל הכלים ובפקדים נפוצים אחרים בצורה טובה, יש צורך להבין את המונח **אוסף** (Collection).

הבנת המושג אוסף

אוסף הוא קבוצת אובייקטים. במקרה של פקד ImageList תמצא אוסף הנקרא ListImages. האובייקטים האגורים באוסף מיוחד זה הם הציורים בפקד ImageList. על ידי טיפול באוסף ListImages בתוכנה תוכל להוסיף, להסיר ולשנות ציורים.

חשוב על אוסף כדבר הדומה, אולם לא זהה בדיוק, למערך. תוכל לגשת אליו בדרך שניגשים למערך, כמו בדוגמת הקוד הבאה אשר הופכת תמונת טופס לציור הראשון האגור בפקד ImageList.

```
Set form1.Picture = ImageList1.ListImages(1).Picture
```

שים לב לכך שהקוד מתייחס לאובייקט מסוים בעזרת האינדקס שלו – במקרה זה, 1. זכור כי אוסף מכיל אובייקטים, בניגוד למערך נתונים המכיל ערכים. לאובייקטים אלה יש סט מאפיינים משלהם.

לאובייקט באוסף של לחצנים יש מאפיין מיוחד: מאפיין **Key**. מאפיין Key של אובייקט מסוג לחצן הוא מחרוזת טקסט אשר ניתן להשתמש בה באותה הצורה שבה משתמשים באינדקס:

```
Set form1.Picture = ImageList1.ListImages("Smiley Face").Picture
```

בעקרון, מאפיין Key באובייקט מסוג לחצן נקבע כאשר האובייקט מתווסף לאוסף הלחצנים באמצעות שיטת ההוספה בפרמטר Key של האוסף.

יצירת סרגל הכלים

כדי ליצור פקד מסוג ToolBar בקוד, יש צורך להגדיר פקד ImageList לפני כן (בסביבת העיצוב או בעזרת קוד) ולמלא אותו בציורים. לאחר מכן תוכל להקצות את פקד ImageList לסרגל הכלים בעזרת פקודה שכזו:

```
Set tbrMain.ImageList = ImageList1
```

אחר כך, תוכל לעצב את אוסף הלחצנים של סרגל הכלים כדי ליצור לחצנים המיועדים לסרגל. בדוגמה הבאה, מתווספים לחצן רווח ולחצן רגיל לפקד סרגל הכלים, tbrMain:

```
tbrMain.Buttons.Add 1, "Sep1",, tbrSeperator, 0
tbrMain.Buttons Add 2 "open", "Open File", tbrDefault, 1
```

בדוגמה זו, נוספו שני אובייקטים מסוג לחצנים באוסף הלחצנים על ידי שימוש בפקודת Add. שני הפרמטרים הראשונים מהווים את האינדקס והמפתח (Key) של הלחצן. שני אלה חייבים להיות ייחודיים. הפרמטר השלישי הוא כותרת הלחצן (Caption). הרביעי (tbrSeperator ו-tbrDefault) הוא קבוע המייצג את סוג הלחצן. לבסוף, מספר אינדקס מתוך רשימת הציורים קובע את הציור המשויך ללחצן.

פקודת Add יכולה לתפקד גם כפונקציה. בצורה זו, היא תיצור אובייקט לחצן חדש ותחזיר את המצביע שלו. תוכל לכתוב את הקוד הקודם גם בצורה כזו:

```
Dim btn As Button

Set btn = tbrMain.Buttons.Add (, "Sep1",, tbrSeperator, 0)

Set btn = tbrMain.Buttons.Add (, "open",, tbrDefault)
Btn.ToolTipText = "Click to open a file"
Btn.Caption = "Open File"
Btn.Image = 1
```

שים לב לכך שמאפיין ToolTipText חייב להתווסף לאובייקט הלחצן מפני שהוא לא נמצא בפרמטרים של פקודת Add. בנוסף, בדוגמה הקודמת, לא הוגדרו מאפייני אינדקס על ידינו ולכן הם הוגדרו בצורה אוטומטית. בעקרון, שיטה זו היא הדרך המועדפת לתכנות מפני שככל שאובייקטים מתווספים ומוסרים מאוסף מסוים, אינדקס האובייקטים האחרים עלול להשתנות. פרמטר Key, לעומת זאת, ישויך תמיד לאותו האובייקט.

תוכל להשתמש במאפיין Key של הלחצן בתוכנית שלך כדי לקבוע איזה לחצן נלחץ. לדוגמה, הנה קוד המיועד לשגרת ButtonClick של סרגל הכלים:

```
Private Sub Toolbar1_ButtonClick(ByVal Button As ComctlLib.Button)

    Select Case Button.Key
        Case "open"
            'Insert open file code here
        Case "save"
            'Insert save file code here
        Case "exit"
            'Insert code to end the program
    End Select

End Sub
```

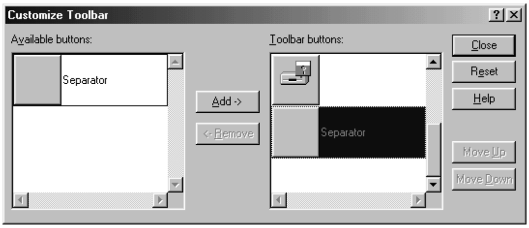
הדוגמה, אשר הובאה כאן, עובדת בצורה טובה עבור לחצנים מסוג tbrDefault. אולם, אם קיימים לחצנים מסוג tbrCheck, יש לשנות את הקוד כך שיבדוק גם את ערך מאפיין Value כדי לדעת מה מצב הלחצן:

```
        Case "boldface"
            If Button.Value = tbrUnpressed then
                'Button is "Up" – Turn bold off
            Else
                'Button is "Down" – Turn bold on
            End If
```

במידה ותקבע את ערך מאפיין MixedState של לחצן כ-True, הלחצן יהיה אפור תמידית, ללא קשר לערך מאפיין Value.

מתן אפשרות למשתמש לעצב את סרגל הכלים

אחד הדברים היפים בפקד מסוג ToolBar היא היכולת לאפשר למשתמש לעצב את הסרגל על פי בחירתו. כאשר מאפיין AllowCustomize נקבע כ-True, יש למשתמש גישה לתיבת הדו-שיח **Customize** על ידי לחיצה כפולה על הסרגל. תיבה זו, הנראית בתרשים 6.17, מאפשרת למשתמש להוסיף לחצנים לסרגל, להסיר לחצנים או להזיז אותם למקום אחר.



תרשים 6.17: מאפיין AllowCustomize מאפשר למשתמש לעצב את סרגל הכלים לפי טעמו, ללא צורך בהוספת קוד נוסף

שימוש בפקד מסוג CoolBar

פקד **CoolBar**, שהוצג קודם לכן בתרשים 6.11, הינו פקד מיוחד המכיל פקדים אחרים אך נראה כמו סרגל כלים. פקד זה אינו מכיל אובייקטים שהם לחצנים מובנים כמו סרגל הכלים הרגיל, אולם משום שהוא משמש כמכולה לאובייקטים, ניתן לשמור בו גם אובייקטים מסוג לחצנים ואף אובייקטים מסוג סרגלי כלים.

פקד CoolBar מכיל אוסף של קבוצות אובייקטים אשר כל אחד מהם הוא מכולה קטנה יותר שהמשתמש יכול לשנות בזמן פעולת התוכנית.

כדי להדגים כיצד פקד CoolBar מסוגל להכיל פקדים אחרים, הבה נרחיב את הדוגמה מסעיף קודם. בצע את הפעולות הבאות:

1. אם לא עשית זאת לפני כן, הוסף פקד מסוג **CoolBar** לארגז הכלים על ידי בחירת **Microsoft Windows Common Controls - 3 6.0 Components** מתיבת הדו-שיח.
2. שנה את ערך מאפיין **Align** בסרגל הכלים שיצרת לפני כן ל- **0 – vbAlignNone**. גרור אותו לשטח פתוח בטופס וכוון את רוחבו כך שיהיה רחב מספיק כדי להציג את הלחצנים שהוספת לפני כן.
3. לחץ לחיצה כפולה על סמל **CoolBar** בארגז הכלים כדי להוסיף פקד **CoolBar** לטופס.
4. לחץ על פקד **CoolBar** כדי לבחור אותו. לחץ על סמל **CommandButton** וצייר לחצן פקודה מעל לפקד **CoolBar**.

5. צור שגרת Click פשוטה עבור הלחצן, והוסף שורת קוד כגון:

```
MsgBox "You clicked Command1".
```

6. לחץ לחיצה ימנית על סרגל הכלים ובחר באפשרות **Cut** כדי להסיר אותו מהטופס ולהציב אותו בלוח של Windows.

7. לחץ לחיצה ימנית על פקד **CoolBar** ובחר באפשרות **Paste**. סרגל הכלים הנמצא בלוח יועבר לתוך מכולת **CoolBar**.

8. לחץ לחיצה ימנית על שטח ריק בפקד **CoolBar** (אל תלחץ על פקד **ToolBar** או על הלחצן). בחר במאפיינים כדי להציג את תיבת הדו-שיח **Property Pages** של הפקד.

9. הכרטיסיה **Bands** של **Property Pages** מכילה מאפיינים עבור כל אחד מחלקי הפקד. חלק 1 אמור להיבחר בתיבת האינדקס בחלק העליון. פתח את רשימת **Child** ובחר **tbrDemo**. פעולה זו מורה לפקד **CoolBar** להציג את **tbrDemo** במיני-מכולה Band 1.

10. שנה את האינדקס ל-2 כדי לבחור את החלק השני. קבע את מאפיין **Child** שלו כ-**Command1** (הלחצן שהוספת לפני כן).

11. לחץ על לחצן **OK** כדי לסגור את **Property Pages**. בזמן שפקד **CoolBar** עדיין נבחר, עבור לחלון המאפיינים ושנה את מאפיין Align שלו ל- **1 – vbAlignTop**.

כאשר סיימת את הצעדים הללו, הפעל את התוכנית וצפה בהתנהגות האובייקטים הנמצאים בפקד CoolBar. תוכל לחקור אותם על ידי הצבת סוגים שונים של אובייקטים בפקד זה בתוכניות שלך.

מכאן...

פרק זה הציג בפניך את היתרונות של יצירת תפריטים וסרגלי כלים בתוכניתיך. תקוותנו היא כי ראית כמה פשוט ליצור את הפריטים הללו. בנוסף, נחשפת למספר נושאים אחרים בפרק זה. כדי לקבל מידע נוסף עליהם, פנה לפרקים הבאים:

- למד כיצד להשתמש במסמכים לשם בניית יישומים בפרק 3 **אבני היסוד של Visual Basic**.
- למד דרך נוספת ליצירת קשר בין התוכנית למשתמש בפרק 7 **שימוש בתיבות דו-שיח לקבלת מידע**.
- למד עוד על שימוש במשתנים לשם אחסון מידע שוטף בתוכניותיך, בפרק 8 **שימוש במשתנים לאחסון מידע**.

7

שימוש בתיבות דו-שיח לקבלת מידע

מה בפרק?

- יידוע המשתמש לגבי מצב התוכנית ופעולותיה
- השגת מידע מן המשתמש
- שימוש בתיבות דו-שיח מובנות
- יצירת תיבות דו-שיח משלך

תיבת דו-שיח (Dialog Box), היא חלון המשמש להצגת/השגת מידע. השם "תיבת דו-שיח" מרמז על כך שמדובר למעשה באמצעי שיחה, או "דיאלוג", עם המשתמש. בדרך כלל מוצגת תיבת הדו-שיח בתוכנית באופן **מודאלי** (Modally) שבו על המשתמש לסגור אותה (או "להשתתף בדו-שיח") בטרם יוכל להמשיך הלאה, לחלקים אחרים בתוכנית. בפרק זה תערוך היכרות עם שתיים מתיבות הדו-שיח המובנות ב-Visual Basic : תיבת ההודעה (Message Box) ו- תיבת הקלט (Input Box). אחר כך תלמד להשתמש בפקד המותאם אישית - CommonDialog, המאפשר לך לשלב בתוכניתך ארבעה סוגים נוספים של תיבות דו-שיח תקניות ("סטנדרטיות"). ובסוף, תערוך היכרות ראשונית עם מספר קווים מנחים, אשר יסייעו לך ביצירת תיבות דו-שיח מבוססות-טופס משלך.

יידוע המשתמש לגבי מצב התוכנית ופעולותיה

חלק חשוב בכל פרויקט תכנות הוא מסירת מידע למשתמש באשר למידת התקדמותה ומצבה של התוכנית. טפסים ופקדים הם אומנם החלק העיקרי בממשק שלך, אך אינם בהכרח האמצעי הטוב ביותר להצגת מידע המחייב את תשומת ליבו המיידית של המשתמש, כגון - הודעות אזהרה וטעות. **תיבת הודעה** (Message Box) היא האמצעי הטוב ביותר להצגת מידע מסוג זה.

הכרת תיבת ההודעה - Message Box

תיבת ההודעה היא טופס פשוט המכיל הודעה ולפחות לחצן פקודה אחד. הלחצן מיועד לאישור ההודעה וסגירתה. הואיל ותיבות הודעה הן חלק אינטגרלי משפת Visual Basic, אינך צריך לדאוג לאופן יצירתן או עיצובן. כדי לראות תיבת הודעה פשוטה, הקלד את שורת הפקודות הבאה בחלון Immediate של Visual Basic (לצורך פתיחת החלון בחר View, Immediate Window):

```
MsgBox "I love Visual Basic!"
```

באמצעות שימוש בפרמטרים אופציונליים, ניתן להוסיף לתיבת ההודעה סמלים ולחצנים שונים המאפשרים דו-שיח עם המשתמש. שורת הקוד הבאה יוצרת את תיבת הדו-שיח המופיעה בתרשים 7.1:

```
MsgBox "Delete record?", vbYesNo + vbExclamation, "Confirm Delete"
```

הערה:

אתה יכול לנסות את הדוגמה בעצמך. הקלד את שורת הקוד בחלון Immediate והקש Enter. פעולה זו תגרום להפעלת הפקודות בדומה להפעלתן בזמן ריצה.



תרשים 7.1: תיבת ההודעה מתקשרת עם המשתמש

אתה יכול לנצל את תיבת ההודעה לשתי מטרות שונות, בהתאם לצרכיך: התיבה יכולה להוות כלי פשוט להצגת מידע, או לחילופין, פונקציה המאפשרת למשתמש להחליט לגבי סוגיה כלשהי בתוכנית. בכל מקרה, יהא עליך להשתמש, באופן כלשהו, בפונקציה MsgBox.

הערה:

פרק זה אינו מבצע אבחנה טרמינולוגית בין **פונקציות** (Functions) ל**הצהרות** (Statements), עם זאת, קיים הבדל בין השימוש ב-MsgBox כפונקציה או כהצהרה. על פי ההגדרה, **פונקציה** משיבה אל התוכנית ערך כלשהו, בעוד ש**הצהרה** לא. גם התחביר המשמש בכתיבת הפרמטרים של פונקציות והצהרות שונה במקצת. פרמטרים של פונקציה יבואו תמיד בתוך סוגריים. לדוגמה - אם אתה מעוניין לצפות בערכים שנקלטו בפונקציית MsgBox הקלד את שורת הקוד הבאה:

```
Print MsgBox ("Delete record?", vbYesNo + vbExclamation, "Confirm Delete")
```

תיבת ההודעה הינה שימושית ביותר, אך יש לה גם מספר מגבלות:

- לא ניתן להכניס לתוכה קלט בצורת טקסט. התיבה מוגבלת להצגת מידע וטיפול במספר מסוים של אפשרויות, המתורגמות ללחצני פקודה.
- ניתן לשבץ בתיבה אחד מתוך ארבעה סמלים מוגדרים מראש, ואחת מתוך שש מערכות מוגדרות מראש של לחצני פקודה. אין אפשרות לשלב בתיבה סמלים או לחצני פקודה משלך.
- תיבת ההודעה עוצרת את המהלך השוטף של התוכנית וממתינה להתייחסות המשתמש. לפיכך, אין היא יכולה לשמש כלי למעקב שוטף אחר התקדמות התוכנית, שכן כל עוד היא מופיעה על המסך, לא ניתן לבצע את חלקי התוכנית האחרים.

הצגת הודעה

הדרך הקלה ביותר ליצור תיבת הודעה עבור התוכנית שלך, היא על ידי שימוש בפונקציית MsgBox כהצהרה, ולא כפונקציה (כלומר מבלי להחזיר ערכים לתוכנית). באופן זה אתה פשוט קובע את נוסח ההודעה אשר אתה מעוניין להציג למשתמש, וקורא לפונקציה MsgBox כדי לבצע את מבוקשך. תיבת ההודעה מכילה, במקרה זה, הודעה כלשהי ולחצן OK בודד, המאפשר למשתמש לאשר את תוכן ההודעה:

```
MsgBox "Record processing is complete."
```

בעת קביעת נוסח ההודעה, אתה יכול להשתמש ב**קבוע מחרוזת** (String Constant) או ב**משתנה מחרוזת** (String Variable). קבוע מחרוזת הוא קטע טקסט מוגדר הנתון בין שני סימני גרשיים, כפי שמציגה דוגמת הקוד האחרונה. משתנה מחרוזת מוצג בדוגמה להלן:

```
sMyText As String

sMyText = "Hello," & vbCrLf & "World"
MsgBox sMyText
```

שים לב לשימוש בקבוע המוגדר מראש - vbCrLf, המייצג את הצירוף line feed/carriage return, במטרה לגרום למילים "Hello" ו-"world" להופיע בשורות נפרדות. שילוב זה הוא שווה ערך לשילוב: chr(10) & chr(13).

תיבת ההודעה, המתקבלת במצב ברירת מחדל, מכילה לחצן OK בודד ומשתמשת בשם הפרויקט שלך ככותרת (כדי להגדיר שם פרויקט משלך בחר באפשרויות **Project**, **Properties**). תוכל "לאבזר" מעט את התיבה באמצעות שני פרמטרים אופציונליים: ארגומנט buttons וארגומנט title. ארגומנט buttons הוא מספר שלם, באמצעותו ניתן לבטא את הסמל שנבחר עבור תיבת ההודעה, את מערכת לחצני הפקודה ואת הלחצן המשמש כברירת מחדל. ארגומנט title הוא מחרוזת טקסט המכילה את שורת הכותרת של תיבת ההודעה. להלן פורמט ההגדרה המלא של תיבת ההודעה:

```
MsgBox (promp [, buttons] [, title] [, helpfile, context])
```

אם אתה מעוניין, תוכל לשלב בתיבת ההודעה אחד מבין ארבעה סמלים אפשריים. בטבלה 7.1 תמצא סיכום קצר של ארבעת הסמלים אשר ניתן לשלב בתיבה, לצד מטרת השימוש בהם.

טבלה 7.1: סמלים המעידים על סוג ההודעה המוצגת בתיבת ההודעה

שם הסמל	שימוש מקובל
הודעה חשובה ביותר	מעיד על התרחשותה של שגיאה חמורה. בדרך כלל התוכנית תיסגר לאחר הופעת הודעה מסוג זה.
אזהרה	מעיד על התרחשותה של שגיאת תוכנית העלולה להוביל לתוצאות בלתי רצויות אם לא תתוקן על ידי המשתמש.
שאילתה	מורה על כך שהתוכנית זקוקה למידע נוסף מן המשתמש בטרם תוכל להמשיך בעיבוד הנתונים.
הודעת מצב	יידוע המשתמש לגבי מצב התוכנית. לעיתים קרובות נעשה שימוש בהודעה מסוג זה לציון סוף תהליכים שונים בתוכנית.

הגדר ערך עבור ארגומנט buttons בפונקציה MsgBox, כדי ש- Visual Basic תדע כי אתה מעוניין לשלב סמל בתיבת ההודעה שלך. ניתן לבחור עבור ארגומנט buttons אחד מבין ארבעת הערכים המובאים בטבלה שלהלן ולבטאו בפונקציה כמספר או כקבוע.

סוג ההודעה	ערך מספרי	קבוע
הודעה חשובה ביותר	16	vbCritical
שאילתה	32	vbQuestion
אזהרה	48	vbExclamation
הודעת מצב	64	vbInformation

טיפ:

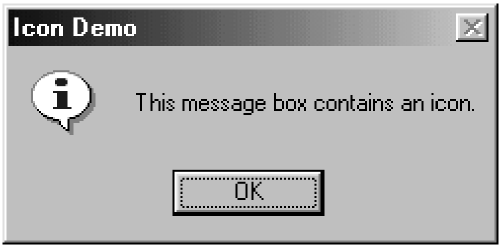
באופן עקרוני, רצוי יותר להשתמש בקבועים (ולא בערכים מספריים) להגדרת ערכי הארגומנטים. שימוש נכון בקבועים יקל על קריאת תוכניתך ויחסוך ממך כאבי ראש מיותרים במקרה ש- Microsoft תחליט לשלב בגרסאות עתידיות, ערכים מספריים חדשים. כמו כן שימוש בקבועים הופך את הקוד לקריא יותר עבור מפתחים אחרים אם וכאשר יצטרכו לתחזק את הקוד.

הערה:

הקבועים המופיעים בטבלה אחראים גם על הצליל המופק על ידי Windows בעת הופעת ההודעה על המסך. תוכל להגדיר צלילים עבור סוגים שונים של הודעות באמצעות לוח הבקרה של Windows.

להדגמת אופן השימוש בסמלים וכותרות בתיבת ההודעה שלך, הקלד את שורת הקוד הבאה, הגורמת ליצירת תיבת ההודעה המוצגת בתרשים 7.2 :

```
MsgBox "This message box contains an icon.", vbInformation, "Icon Demo"
```



תרשים 7.2: השתמש בכותרות וסמלים כדי להבליט את מאפייני ההודעה

ערכים המוחזרים על ידי פונקציית MsgBox

כפי שכבר למדת, נוח להשתמש בפונקציה MsgBox כדי להתריע בפני בעיות שונות או להודיע למשתמש שעליו לבצע מהלך מסוים. אולם, כאשר נדרשת החלטה מצד המשתמש בעניין כלשהו, עליך להשתמש בערכים המוחזרים על ידי הפונקציה MsgBox. קיימים שני הבדלים עיקריים בעת השימוש בפונקציה באופן זה : עליך להתייחס לערך המוחזר, לרוב על ידי הגדרתו כמשתנה, ולתחם את הארגומנטים של הפונקציה בתוך סוגריים. הערך המוחזר מייצג את הלחצן אשר נבחר על ידי המשתמש. שורת הקוד הבאה מדגימה כיצד להגדיר את הערך המוחזר כמשתנה, כדי שניתן יהיה לעשות בו שימוש בהמשך :

```
nResult = MsgBox ("The printer is not responding", vbRetryCancel, _
   "Printer Error!")
```

בהמשך התוכנית תוכל לבדוק את הערך שנקלט במשתנה nResult באמצעות משפט If למשל, ולהגיב בהתאם לתוצאה שתתקבל.

הערה:

Visual Basic מתריעה על כל שימוש ב-MsgBox כפונקציה במידה ולא הוקצה משתנה לקליטת הערך החוזר (כלל זה תקף עבור כל פונקציה).

בחירת מערכת לחצני הפקודה

בתיבת ההודעה ניתן לשלב אחת משש מערכות הלחצנים הבאות:

1. **OK**. הצגת לחצן בודד בשם OK. הלחצן משמש פשוט כדי להבטיח שהמשתמש יתייחס להודעה בטרם תמשיך התוכנית בפעולתה.

- **OK, Cancel**. הצגת צמד לחצנים המאפשרים למשתמש לבחור בין אישור ההודעה לביטול המהלך.
- **Abort, Retry, Ignore**. הצגת מערכת של שלושה לחצנים המשמשת בדרך כלל בהודעת טעות. באפשרות המשתמש לבחור בין ביטול המהלך, התחלתו מחדש או המשך ביצועו תוך התעלמות מן הטעות שנתגלתה בו.
- **Yes, No, Cancel**. מערכת של שלושה לחצנים המשמשת בדרך כלל בהודעת שאלה. המשתמש יכול להשיב לשאלה ב"כן" או "לא", או לחילופין, לבטל את המהלך כולו.
- **Yes, No**. הצגת צמד לחצנים עבור שאלות "כן-לא" פשוטות.
- **Retry, Cancel**. הצגת צמד לחצנים המאפשרים למשתמש להתחיל במהלך מחדש או לבטלו. מקובל להשתמש במערכת לחצנים זו כאשר המדפסת אינה מגיבה. במקרה זה, המשתמש יוכל להמשיך במהלך לאחר תיקון התקלה, או לחילופין, לבטל את ההדפסה.

כדי לשלב בתיבת ההודעה את מערכת הלחצנים המתאימה, הגדר ערך עבור ארגומנט buttons בפונקציה MsgBox. בטבלה 7.2 תוכל למצוא את הערכים המשויכים לכל אחת ממערכות הלחצנים האפשריות.

טבלה 7.2: בחר עבור הארגומנט buttons את אחד הערכים בטבלה כדי לשלב בתיבת ההודעה את מערכת הלחצנים המתאימה

מערכת לחצנים	ערך	קבוע
OK	0	vbOKOnly
OK, Cancel	1	vbOKCancel
Abort, Retry, Ignore	2	vbAbortRetryIgnore
Yes, No, Cancel	3	vbYesNoCancel
Yes, No	4	vbYesNo
Retry, Cancel	5	vbRetryCancel

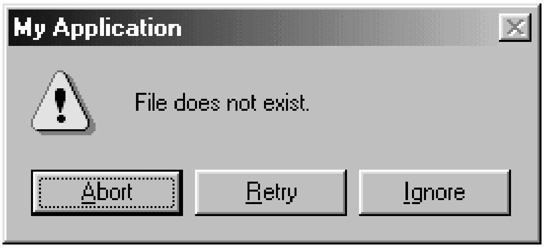
הואיל וארגומנט buttons אחראי הן לסמל והן ללחצני הפקודה של תיבת ההודעה, אתה בוודאי שואל את עצמך האם וכיצד ניתן להגדיר את שני הערכים במקביל. הגדרתם של קבוע הסמל וקבוע לחצני הפקודה, במקביל, יוצרת ערך מיוחד. פונקציית

MsgBox יודעת לפרק את הערך המיוחד ולחלקו לערכים עצמאיים, המייצגים כל רכיב של תיבת ההודעה בנפרד. שורות הקוד הבאות מראות כיצד ניתן לשלב קבוע סמל וקבוע לחצני פקודה בהודעת אזהרה, המאפשרת למשתמש לבחור אחת מבין שלוש פעולות אפשריות. תוצאת הקוד מופיעה בתרשים 7.3:

```
nOptval = vbExclamation + vbAbortRetryIgnore
nReVal = MsgBox ("File does not exist. ", nOptval, "My Application")
```

הערה:

אם אתה מעוניין לשלב בתיבת ההודעה גם לחצן Help, הוסף להגדרת הפונקציה את הקבוע vbMsgBoxHelpButton, אשר מקבל ערך של 16384 עבור כל מערכת לחצנים אליה הוא מצורף.



תרשים 7.3: ארגומנט buttons אחראי הן לסמל והן ללחצני הפקודה המוצגים על ידי הפונקציה MsgBox

הגדרת לחצן ברירת מחדל

כאשר תיבת ההודעה מכילה יותר מלחצן אחד, באפשרותך להגדיר אחד מן הלחצנים כברירת המחדל. **לחצן ברירת המחדל** (Default Button) הוא זה המצוי במיקוד בעת הצגת התיבה. בדרך כלל, ייבחר לצורך העניין הלחצן אשר קיימת סבירות גבוהה כי בו יבחר המשתמש, לפיכך, כל שעליו לעשות הוא להקיש על מקש Enter. לדוגמה, אם יצרת תיבת הודעה אשר באמצעותה מתבקש המשתמש לאשר מחיקת רשומה, מוטב יהיה אם תגדיר, כברירת מחדל, לחצן המונע את מחיקת הרשומה. באופן זה אתה מבטיח כי מחיקת הרשומה תיעשה מתוך בחירה מודעת של המשתמש.

כדי להגדיר את לחצן ברירת המחדל, הוסף קבוע נוסף להגדרת הפונקציה MsgBox. תוכל לבחור אחד מבין ארבעה ערכי ברירת מחדל אפשריים, המוצגים בטבלה הבאה:

לחצן ברירת מחדל	ערך	קבוע
ראשון	0	vbDefaultButton1
שני	256	vbDefaultButton2
שלישי	512	vbDefaultButton3
רביעי	768	vbDefaultButton4

בסיכומו של דבר, ניתן לשלב בתיבת ההודעה שבעה סוגי לחצנים שונים, בכפוף, כמובן, לצירופי הלחצנים האפשריים. כל אחד מן הלחצנים מחזיר ערך מספרי שונה, באמצעותו מזהה התוכנית את הלחצן שנבחר ומגיבה בהתאם (ראה טבלה 7.3):

טבלה 7.3: ערכים מוחזרים המהווים אינדיקציה לבחירת המשתמש.

לחצן	ערך	קבוע
OK	1	vbOK
Cancel	2	vbCancel
Abort	3	vbAbort
Retry	4	vbRetry
Ignore	5	vbIgnore
Yes	6	vbYes
No	7	vbNo

כאמור, במהלך כתיבת הקוד, רצוי לעשות שימוש בקבועים ולא בערכים המספריים של הלחצנים. כאשר ייוודע לך באיזה לחצן בחר המשתמש, תוכל להשתמש במידע במסגרת התוכנית שלך.

יצירת תיבת הודעה מודאלית (Modal Message Box)

עתה, נותר לנו לכסות עוד עניין אחד לשם מיצוי הפוטנציאל הטמון בארגומנט buttons של הפונקציה MsgBox. באמצעות ארגומנט buttons, הינך יכול להגדיר עבור תיבת ההודעה **מודאליות יישום** (Application Modal) או **מודאליות מערכת** (System Modal). בעת הופעת חלון מודאלי על גבי המסך, לא ניתן להמשיך בתוכנית, אלא לאחר סגירתו. לפיכך, אם תגדיר עבור תיבת ההודעה מודאליות של מערכת, יהיה על המשתמש ללחוץ על התיבה בטרם יוכל לבצע איזושהי פעולה עם המחשב כולו. ברור על כן, שיש להשתמש במודאליות של מערכת בזהירות רבה. במצב ברירת מחדל מוגדרת עבור תיבת ההודעה מודאליות של תוכנית בלבד, כך שאין היא מגבילה את המשך עבודה עם תוכניות אחרות הפועלות אותה שעה על גבי המערכת.

הוספת הקבוע vbApplicationModal, שהינו בעל ערך 0, לארגומנט buttons תגרום לפונקציה MsgBox ליצור תיבת הודעה בעלת מודאליות של תוכנית (זהו כאמור גם מצב ברירת המחדל ומתקיים גם ללא הוספת קבוע כלשהו לארגומנט). הוספת הקבוע vbSystemModal, שהינו בעל ערך 4096, תגרום ליצירת תיבת הודעה בעלת מודאליות של מערכת.

הצגת פעולת הפונקציה MsgBox

כדי להדגים את החומר הנלמד עד כה בפרק זה, נבנה תוכנית פשוטה המגדירה תיבת הודעה המבקשת מן המשתמש לאשר כי ברצונו לצאת מן התוכנית בה הוא נמצא. בצע את הצעדים הבאים:

1. הפעל את **Visual Basic** אם צריך, והתחל פרויקט חדש מסוג **Standard EXE**.
2. הוסף לחצן פקודה בודד ל -Form1. היכנס לרשימת המאפיינים של הלחצן, הקלד **cmdExit** במאפיין **Name** (שם) ו- **Exit** במאפיין **Caption** (כותרת).
3. כתוב את משפטי הקוד הבאים לשגרת האירוע ()cmdExit_Click:

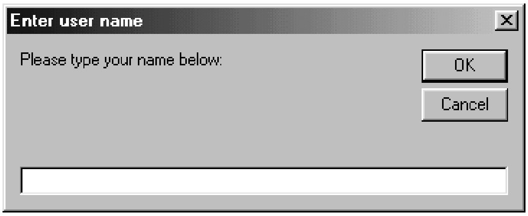
```
Dim sMsg As String
Dim nButtons As Integer
Dim nResult As Integer
sMsg = "Are you sure you want to exit?"
nButtons = vbYesNo + vbQuestion
nResult = Msgbox (sMsg, nButtons, "My program")
if nResult = vbYes Then
   End
End If
```

4. הרץ את התוכנית ולחץ על לחצן **Exit**. כעת אתה נשאל האם אתה בטוח שברצונך לצאת מן התוכנית. לחץ על לחצן **No** ושים לב כי התוכנית אינה נסגרת.

השגת מידע מהמשתמש

פעמים רבות במהלך התוכנית קורה שאתה נזקק לפיסות מידע בודדות מן המשתמש, למשל שם, שם קובץ או מספר כלשהו למטרות שונות. תיבת ההודעה אומנם נותנת למשתמש לבחור בין אפשרויות שונות, אך אינה מאפשרת לו להקליד טקסט כלשהו בתגובה להודעה. לפיכך, עליך להשתמש באמצעים אחרים כדי להשיג מידע מסוג זה. Visual basic מציעה תיבת דו-שיח המיועדת בדיוק למטרה זאת, זוהי **תיבת קלט** (Input Box).

תיבת הקלט מורכבת מהודעה, הרומזת למשתמש מהם הנתונים הדרושים, מתיבת טקסט, לתוכה יכול המשתמש להקליד את הנתונים ומשני לחצני פקודה - OK ו-Cancel, באמצעותם ניתן לאשר או לבטל את המהלך כולו. תרשים 7.4 מציג דוגמה של תיבת הקלט.



תרשים 7.4: תיבת הקלט מאפשרת למשתמש להכניס פיסת מידע בודדת בתגובה להודעה המוצגת בתיבה

הגדרת הפונקציה InputBox

מבחינה טכנית, הגדרת תיבת הקלט דומה מאוד להגדרת תיבת הודעה עם **ערך מוחזר**. ניתן להצמיד את הערך המוחזר מתיבת הקלט למשתנה כלשהו כדי לעשות בו שימוש מאוחר יותר בתוכנית, לקבוע את לשון ההודעה שתוצג בתיבה, ולהגדיר כאופציה, את כותרת התיבה ואת לחצן ברירת המחדל. להלן דוגמת קוד המשמש להגדרת הפונקציה InputBox :

```
Dim sMsg as String, sUserName As String
sMsg = "Please type your name below:"
sUserName = InputBox (sMsg, "Enter user name", "Anonymous")
```

בדוגמה לעיל, המידע המוחזר על ידי פונקציית InputBox מאוחסן במשתנה sUserName. הארגומנט הראשון הוא ארגומנט ההודעה (Prompt argument), המטפל בהודעה שמטרתה להבהיר למשתמש מהו המידע שעליו לספק לתוכנית. בדומה לתיבת ההודעה, גם כאן יכולה ההודעה להכיל עד 1024 תווים בטרם תיקטע. גלישת שורות מבוצעת באופן אוטומטי, כדי להתאים את ההודעה לגודל התיבה. בנוסף, אתה יכול להכניס גם כאן את הצירוף line feed\carriage return (vbCrlf), כדי לגרום להודעה להתפרש על פני מספר שורות או כדי להפריד שורות לצורך הדגשה.

הדוגמה ממשיכה בפרמטרים האופציונליים הכוללים את ארגומנט הכותרת (Title Argument), המטפל בטקסט המופיע בשורת הכותרת של התיבה וארגומנט ברירת המחדל (Default Argument) אשר במידה ונעשה בו שימוש, יגרום להצגת ערך התחלתי מוגדר בתיבת הטקסט. המשתמש יכול לקבל את הערך, לשנותו או למחוק אותו ולהכניס במקומו ערך חדש לחלוטין.

הגדרה מינימאלית של תיבת הקלט מכילה ארגומנט הודעה בלבד, כפי שמוצג להלן :

```
sReturnVal = InputBox ("How’s the weather?")
```

בנוסף לפרמטרים שבאמצעותם ניתן להגדיר שורת כותרת וערך ברירת מחדל לתיבת הטקסט, קיימים פרמטרים אופציונליים נוספים המאפשרים להגדיר את מיקומה הרצוי של התיבה על המסך, ואת קובץ העזרה בו ייעשה שימוש במידת הצורך. הגדרה מלאה של פונקציית InputBox תוכל למצוא במערכת העזרה של Visaul Basic.

הערה:

תיבת הקלט עושה שימוש בשני לחצנים קבועים - OK ו-Cancel. בשונה מפונקציית MsgBox, לא ניתן להחליף לחצנים אלה במערכת לחצנים שונה.

ערכים המוחזרים על ידי פונקציית InputBox

בעת הופעת תיבת הקלט על המסך, יכול המשתמש להקליד עד 255 תווים במקום המיועד לכך, הדומה במאפייניו לתיבת טקסט רגילה. במידה ומספר התווים עולה על 255, הטקסט נגלל שמאלה, הפוך מכיוון הכתיבה. בתום ההקלדה, יכול המשתמש לבחור בלחצן OK או בלחצן Cancel. בחירה בלחצן OK (אישור הפעולה) גורמת לתיבת הקלט להחזיר לתוכנית את מה שנקלט בתיבת הטקסט, הן אם מדובר בערך ברירת המחדל שהוגדר על ידי פונקציית InputBox, והן אם מדובר במידע חדש לגמרי, שהוכנס על ידי המשתמש. בחירה בלחצן Cancel גורמת לתיבת הקלט להחזיר לתוכנית מחרוזת ריקה, מבלי להתחשב במידע שנמצא בתיבת הטקסט.

כדי שניתן יהיה לעשות שימוש במידע שהוכנס לתיבת הקלט, יש לוודא כי הוא עומד בדרישות התוכנית. תחילה עליך לבדוק האם המשתמש אכן הכניס מידע ובחר בלחצן OK. עשה זאת באמצעות הפונקציה Len, אשר תפקידה לבדוק את אורך המחרוזת המוחזרת. אם אורך המחרוזת הוא אפס, הרי הדבר מעיד כי המשתמש בחר בלחצן Cancel, או השאיר את תיבת הטקסט ריקה. אם אורך המחרוזת גדול מאפס, הרי זה סימן שהמשתמש הקליד ערך כלשהו בתיבת הטקסט. כדי לבחון את אופן פעולת הפונקציה Len, הקלד כל אחת משורות הקוד הבאות ב- Immediate Window ושים לב לתוצאות השונות שתתקבלנה:

```
print Len ("Hello")
print Len (" ")
```

בנוסף לכך, עליך לוודא כי הערך המוחזר הוא מן הסוג הדרוש. אם אתה מצפה לערך מספרי, כדי להשוותו מאוחר יותר לערך מספרי אחר על ידי משפט If, עליך להציג הודעת טעות במידה והמשתמש הכניס אותיות. השתמש בפונקציה Val כדי לוודא שהערך המוחזר הוא ערך מספרי. הפונקציה Val משמשת לבדיקת הערך המספרי של מחרוזת. אם המחרוזת מתחילה במספר, הפונקציה Val מחזירה את אותו מספר. בכל מקרה אחר, מחזירה הפונקציה את הערך 0. כדי להבין טוב יותר את הפוטנציאל הטמון בפונקציה Val, הקלד את שורות הקוד הבאות ב- Immediate Window, וראה מה התוצאות שתתקבלנה:

```
print Val ("Hello")
print Val ("50 ways to leave your lover")
print Val ("100 and 1 make 101")
```

הקוד שלהלן מדגים תהליכי עיבוד נוספים של הערך המוחזר על ידי תיבת הקלט, תוך שימוש בפונקציות Val ו-Len:

```
Dim stInputVal As String
stInputVal = InputBox ("Enter your age")
If Len (stInputVal) = 0 Then
  MsgBox "No age was selected"
Else
  If Val (stInputVal) = 0 Then
     MsgBox "You entered an invalid age."
  Else
       MsgBox "Congratulations for surviving this long!"
  End If
End If
```

שימוש בתיבות דו-שיח נפוצות

בסעיפים הקודמים למדת מהי תיבת דו-שיח, וערכת היכרות עם צמד תיבות דו-שיח שכיחות. בסעיפים הבאים תערוך היכרות עם **פקד CommonDialog** של Microsoft, באמצעותו תוכל לשלב בתוכניתך תיבות דו-שיח נוספות, שתסייענה למשתמש בהגדרת שמות קבצים, בחירת גופנים וצבעים ושליטה במדפסת. נוסף על עובדת היותן קלות להגדרה ולשימוש, טמון יתרונן האמיתי של תיבות הדו-שיח בכך שהן מוכרות למשתמש מ-Windows עצמה.

שימושים כלליים בפקד CommonDialog

פקד CommonDialog בודד מקנה לך נגישות לתיבות הדו-שיח הנפוצות הבאות:

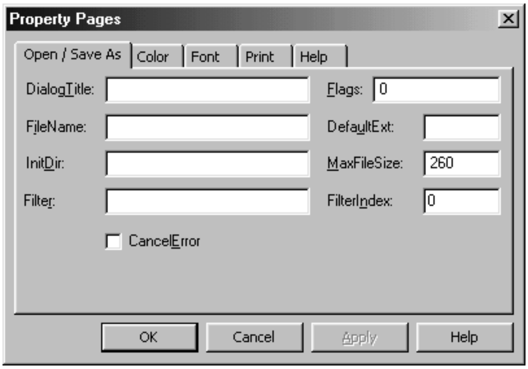
- **Open**. מאפשרת למשתמש להצביע על שם ומיקום הקובץ לשם פתיחתו.
- **Save As**. מאפשרת למשתמש להגדיר שם קובץ ומיקום לשמירת מידע.
- **Font**. משמשת להגדרת הגופן הבסיסי ותכונות הגופן.
- **Color**. משמשת להגדרת צבע במסגרת התוכנית. המשתמש יכול לבחור צבע מתוך טבלת הצבעים הבסיסית, או ליצור צבע מותאם אישית.
- **Print**. משמשת לבחירת מדפסת ולהגדרת חלק מתכונותיה.
- **Help**. מעבירה את המשתמש אל מערכת העזרה של Windows.

על אף שפקד CommonDialog מהווה חלק אינטגרלי מ- Visual Basic, אין הוא מופיע בין הפקדים שבארגז הכלים במצב ברירת מחדל. כדי שתוכל לעשות שימוש בפקד זה, יהיה עליך תחילה להוסיף אותו לפרויקט שלך (ולארגז הכלים) מתוך תיבת הדו-שיח **Components**. תוכל לגשת לתיבה זו באמצעות בחירה ב-**Project Components**.

כעת, היכנס לכרטיסיית **Controls**, בחר באפשרות **Microsoft Common Dialog Control 6.0** ולחץ על לחצן **OK**.

לאחר הוספת פקד **CommonDialog** לארגז הכלים, תוכל להוסיפו לטופס על ידי סימון וציור, בדומה לשאר הפקדים של Visual Basic. פעולה זו תגרום להופעת סמל הפקד על גבי הטופס, זאת מאחר והפקד עצמו אינו נראה לעין בעת הפעלת התוכנית.

הסעיפים הבאים עוסקים בתיבות הדו-שיח השונות אשר ניתן ליצור באמצעות הפקד CommonDialog. עבור כל אחת מן התיבות עליך להגדיר חלק מן המאפיינים של הפקד. תוכל לעשות זאת באמצעות חלון **Properties** (מאפיינים) או על ידי שימוש בתיבת הדו-שיח **Property Pages**. תיבת הדו-שיח Property Pages של הפקד מהווה כלי נוח להגדרת התכונות המיוחדות לכל תיבה ותיבה (ראה תרשים 7.5). ניתן לגשת לתיבת דו-שיח זו על ידי כניסה לחלון Properties של הפקד, ולחיצה כפולה על המאפיין **Custom Property**, או על ידי לחיצה ימנית על הפקד עצמו ובחירה באפשרות **Properties** מתוך התפריט שיופיע על המסך.



תרשים 7.5: תיבת הדו-שיח Property Pages מאפשרת לך להגדיר את מאפייני תיבות הדו-שיח שלך בקלות רבה

בחינת ביצועי הפקד CommonDialog

בסעיף זה נבחן סגנונות שונים של תיבות הדו-שיח הנפוצות. בצע את ההכנות הבאות:

1. הפעל את **Visual Basic** אם צריך, והתחל פרויקט חדש מסוג **Standard EXE**.
2. היכנס לשורת התפריטים ובחר **Project**, **Components**.
3. לחץ על תיבת הסימון שליד הפקד **Microsoft Common Dialog Control 6.0**, וודא שהופיע הסימן הדרוש בתיבה.
4. לחץ **OK** כדי לסגור את תיבת הדו-שיח **Components**.
5. כעת בחר את פקד **CommonDialog** מארגז הכלים והוסף אותו לטופס **Form1**.
6. הקלד **cdlTest** במאפיין **Name** של הפקד.
7. הוסף לחצן פקודה ל-**Form1**. הקלד **cmdTest** במאפיין **Name** של הלחצן ו- **Test Common Dialog** במאפיין **Caption** (כותרת).

ההגדרות שיצרת, תשמשנה כמעטפת לבחינת אפשרויות העבודה השונות עם פקד CommonDialog. כשתתבקש, הכנס את הקוד הדרוש לשגרת האירוע של cmdTest.

תיבות דו-שיח המטפלות בקבצים

אחד המטרות העיקריות לשימוש בפקד CommonDialog היא קליטת שמות קבצים מן המשתמש. ניתן ליצור באמצעות הפקד תיבת דו-שיח לטיפול בקבצים בעלת שני מופעים אפשריים: **Open** (פתח קובץ) ו- **Save As** (שמור כ..). מצב Open מאפשר למשתמש להגדיר שם קובץ ומיקומו לשם פתיחתו. מצב Save As מאפשר למשתמש להגדיר שם ומיקום לשמירת קובץ.

תיבות הדו-שיח Open ו- Save As

תיבות הדו-שיח, באמצעותן מבוצעות פעולות Open ו- Save As, זהות בצורתן. תרשים 7.6 מראה את תיבת הדו-שיח Open. להלן הרכיבים העיקריים:

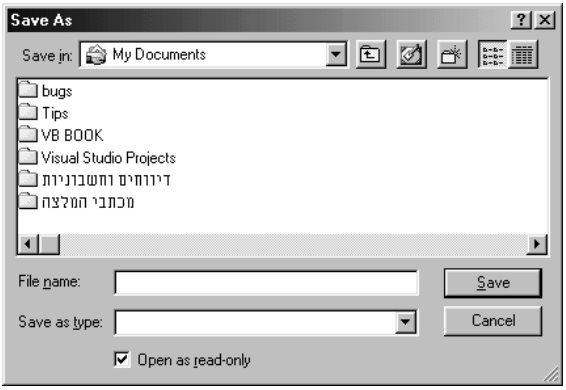
- **רשימת כוננים\תיקיות**. תיבה משולבת בה מופיע שם התיקיה הנוכחית. אם התיקיה הנוכחית היא תיקיית השורש (\), יופיע בתיבה המשולבת שם הכונן הנוכחי. השתמש בתיבה המשולבת (Combo Box) ובלחצני הניווט (Navigation Buttons) למעבר בין רמות תיקיה שונות, כמקובל בסייר Windows.

- **רשימת בחירת קובץ\תיקיה**. תיבת רשימה בה מופיעים הפריטים הנמצאים רמה אחת מתחת לפריטים שברשימת הכוננים\תיקיות. כדי לבחור פריט מן הרשימה, לחץ עליו לחיצה כפולה, או סמן אותו והקש Enter. בחירת תיקיה גורמת לעדכון התצוגה ולהופעת הקבצים ו/או תיקיות המשנה הנכללים בתיקיה הנבחרת. בחירת קובץ גורמת לסגירת תיבת הדו-שיח ופתיחת הקובץ.

- **תיבת שם קובץ**. תיבת טקסט המאפשרת למשתמש להגדיר באופן ידני שם תיקיה או קובץ. הקלדת שם קובץ גורמת לסגירתה של תיבת הדו-שיח. הקלדת נתיב המוביל לתיקיה כלשהי (לדוגמה C:\Data\Word) גורמת להצגת תיקיות המשנה ו\או הקבצים של אותה תיקיה ברשימת בחירת קובץ\תיקיה. כמו כן, לחיצה בודדת על קובץ כלשהו ברשימת קובץ/תיקיה, גורמת להופעתו בתיבת שם קובץ.

- **תיבת סוג קובץ**. תיבת רשימה המאפשרת למשתמש להגדיר את סוג הקבצים שיופיעו ברשימת קובץ/תיקיה. סוג הקובץ נקבע על ידי הסיומת המצורפת לשם הקובץ. סוגי הקבצים האפשריים נקבעים באמצעות המאפיין Filter (מסנן) של הפקד CommonDialog.

- **לחצני סרגל כלים ולחצני פקודה**. הלחצנים המופיעים בפינה הימנית-עליונה של התיבה מאפשרים למשתמש לעלות רמת תיקיה אחת, ליצור תיקיה חדשה או לשנות את מאפייני התצוגה ברשימת קובץ/תיקיה, ממצב רשימה למצב פרטי קובץ ולהפך. בעזרת הלחצנים Open ו-Cancel, המופיעים בפינה הימנית-תחתונה, יכול המשתמש להורות לתוכנית לפתוח את הקובץ המסומן, או לבטל את המהלך כולו.



תרשים 7.6: תיבות הקבצים Open ו- Save As חולקות רכיבים משותפים רבים

פתיחה ושמירת קבצים

כדי לפתוח קובץ קיים השתמש בשיטה ShowOpen של פקד CommomDialog (שיטה זו גורמת להצגת תיבת הדו-שיח שבתרשים 7.6). עליך להגדיר את שם הפקד CommonDialog ואת שם השיטה. לשם הדגמה, הקלד את שורות הקוד הבאות לשגרת האירוע cmdTest_Click:

```
cdlTest.Showopen
Msgbox "You Selected " & cdlTest.Filename & " to be opened."
```

לאחר ביצוע קוד זה, יכולה התוכנית שלך להשתמש בשם הקובץ שנבחר דרך מאפיין FileName של פקד CommonDialog.

פתיחה או שמירה של קובץ באמצעות פקד CommonDialog כרוכות למעשה בביצוע פעולות דומות. השיטה המשמשת להפעלת תיבת הדו-שיח Save As נקראת ShowSave. קיימים מספר הבדלים עדינים בין תיבות הדו-שיח Open ו-Save, למשל בכותרת המופיעות בראש התיבה ועל גבי לחצני הפקודות.

כדי לבחון את פעולת השיטה ShowSave של הפקד CommonDialog, החלף את הקוד בשגרת האירוע cmdText_click בשורות הקוד הבאות.

בתקליטור:

ראה תוכנית Chapter7.VBP בתקליטור.

```
cdlTest.ShowSave
Msgbox "You Selected " & cdlTest.FileName & "to be saved. "
```

הערה:

תיבות הדו-שיח Open ו- Save As אינן באמת פותחות או שומרות קבצים. הן רק מקבלות מן המשתמש מידע באשר לשמם ומיקומם של הקבצים המיועדים לפתיחה או שמירה. תוכניתך היא זו הצריכה לדאוג להשלמת הפעולה.

הגדרת סוגי קבצים באמצעות Filter (מסנן)

בעת השימוש בפקד CommonDialog, ייתכן שתרצה להציג רק קבצים מסוג מסוים בתיבות הדו-שיח. לדוגמה, אם תוכניתך עוסקת בקבצים של תוכנת Microsoft Excel, תרצה שבתיבות הדו-שיח יוצגו קבצים בעלי סיומת XLS. ולא קבצים בעלי סיומת BAT. תוכל להגביל (או "לסנן") את הקבצים בתיבת הדו-שיח באמצעות רכיב אופציונלי של Visual Basic המכונה **Filter** (מסנן).

הגדרת המסנן מתבצעת בזמן עיצוב, באמצעות חלון **Properties** (מאפיינים) או באמצעות תיבת הדו-שיח **Property Pages**, ובזמן ריצה באמצעות פקודה מיוחדת המשולבת בקוד. הפילטר הינו מחרוזת טקסט הכוללת את סוג הקובץ והסיומת שלו. הדבר מחייב פורמט מיוחד כמוצג להלן:

```
cdlTest.Filter = "Word Documents (*.doc) | *.doc"
```

הסימן האנכי (|) שמופיע בקוד מכונה "סימן מקטרת" (**pipe symbol**) ועליו להופיע בהגדרת המסנן. לפני סימן pipe מופיע תיאור קצר של סוג הקובץ, במקרה זה - Word Document (*.doc). לאחר סימן pipe מופיע המסנן הלכה למעשה. המסנן מורכב בדרך כלל מכוכבית (*) ואחריה נקודה (.) וכן סיומת הקבצים שברצונך להציג, לדוגמה: *.doc, *.txt או *.* .

כשאתה מגדיר את המסנן באמצעות פקודה בקוד, עליך לתחם אותו בין סימני מרכאות, בדומה לכל מחרוזת אחרת. אין צורך בסימני מרכאות בעת הגדרת המסנן בזמן עיצוב, דרך תיבת הדו-שיח **Propety Pages**.

ניתן להגדיר זוגות רבים של "תיאור | מסנן" (description | filter) במאפיין מסנן בודד. יש להפריד כל זוג על ידי סימן pipe, כפי שניתן לראות בדוגמה הבאה:

```
cdlTest.Filter = ″Text Files | *.txt|All Files|*.*″
```

כדי לבחון את פעולת המסנן, החלף את הקוד בשיגרה cmdTest_Click של פרויקט הדוגמה בשורות הקוד הבאות:

```
cdlTest.Filter = ″Text Files (*.txt) |*.txt|All Files|*.*″
cdlTest.ShowOpen
```

בדוגמה לעיל, המסנן גורם להופעת קבצי טקסט בלבד בתיבת הדו-שיח של פקד CommonDialog. ניתן להחליף את המסנן המקורי ולבחור מסנן חדש (במקרה זה - All Files) מתוך תיבת הרשימה הנפתחת Type (סוג).

התאמה אישית של תיבות הדו-שיח על ידי שימוש בדגלונים

מאפיין חשוב נוסף של תיבות הדו-שיח הם ה**דגלונים** (Flags). הדגלונים מוגדרים על ידי שימוש בקבוע או מערכת של קבועים, בדומה לפרמטר **Buttons** לדוגמה: **vbOKOnly** או **vbYesNo** של הפונקציה MsgBox. רשימה מלאה של קבועים האפשריים תוכל למצוא במערכת העזרה של Visual Basic. לדוגמה, אם אינך מעוניין

שתיבת הסימון Open as Read Only תופיע בתוכניתך, תוכל לגרום להסתרתה על ידי דגלון מיוחד, באופן הבא:

```
cdlTest.Flags = cdlOFNHideReadOnly
cdlTest.InitDir = "C:\Windows".
cdlTest.ShowOpen
```

שים לב גם לשימוש במאפיין InitDir המאתחל את תיבת הדו-שיח בספריה מסוימת.

אזהרה:

דגלונים, מסננים ומאפיינים אחרים המשפיעים על פקד CommonDialog, יש להגדיר לפני הצגתו בתוכנית.

תיבת הדו-שיח Font (גופן)

ניתן להגדיר את פקד CommonDialog כך שיציג את תיבת הדו-שיח Font. הפעולה הינה נוחה וקלה, בדומה לתיבות הקבצים. למעשה, פקד CommonDialog אחד, המוצב על גבי הטופס, יכול לשמש לטיפול בקבצים, גופנים, צבעים ופונקציות מדפסת, זאת, על ידי הגדרת המאפיינים כל פעם מחדש וקריאה לשיטה המתאימה.

השלב הראשון ביצירת תיבת הדו-שיח Font, הוא מתן ערך למאפיין Flag. מאפיין זה, אחראי בין היתר להורות לפקד CommonDialog האם להציג בתיבת הדו-שיח - גופני מסך, גופני מדפסת או את שניהם. הקבועים האפשריים מופיעים בטבלה הבאה:

הגדרת גופן	קבוע	ערך
גופני מסך	cdlCFScreenFonts	1
גופני מדפסת	cdlCFPrinterFonts	2
גופני מסך ומדפסת	cdlCFBoth	3

אזהרה:

אי מתן ערך למאפיין Flag יגרור הודעת טעות המתריעה על כך שלא הוגדרו גופנים עבור תיבת Font.

ניתן להגדיר ערך עבור מאפיין Flag מסביבת הפיתוח, דרך חלון Properties או תיבת Property Pages, או מתוך הקוד באמצעות פקודת משימה מתאימה. לאחר הגדרת מאפיין Flag, תוכל להשתמש בשיטה ShowFont של פקד CommonDialog, כדי לקרוא לתיבת הדו-שיח Font מתוך התוכנית. עם סגירתה של תיבת Font, יופיעו הגדרות הגופנים שבחר המשתמש, בחלון המאפיינים של הפקד CommonDialog.

הצעדים הבאים מאפשרים לך להתאים את תוכנית הדוגמה שיצרת בשלב מוקדם יותר של פרק זה, לצורך מעקב אחר התהליך תוך כדי התרחשותו:

1. אם התוכנית עדיין פועלת כתוצאה מבדיקות קודמות, עצור אותה.
2. הוסף פקד **Label** ל-**Form1**. הכנס את הערך **lblTest** למאפיין **Name** של הפקד ואת הערך **This is a test** למאפיין **Caption**.
3. הכנס את הערך **True** למאפיין **AutoSize** של הפקד.
4. החלף את הקוד בשגרת האירוע Click של cmdTest בשורות הקוד הבאות:

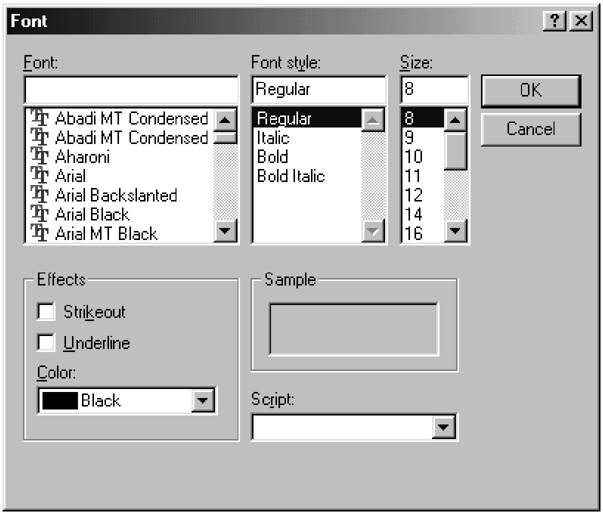
```
cdlTest.Flags = cdlCFBoth + cdlCFEffects
cdlTest.ShowFont
lblTest.Font.Name = cdlTest.FontName
lblTest.Font.Size = cdlTest.FontSize
lblTest.Font.Bold = cdlTest.FontBold
```

5. הפעל את היישום. בעת לחיצה על לחצן **cmdTest** אמורה להופיע על המסך תיבת הדו-שיח **Font**. שנה את גופן ברירת המחדל לגופן שונה בתכלית, לחץ על לחצן **OK** ושים לב כי הגופן של lblTest השתנה לגופן החדש שבחרת.

שני דברים בקוד לעיל דורשים התייחסות מיוחדת: ראשית - הדגלון הנוסף, cdlCEFffects, אשר מאפשר למשתמש להגדיר תכונות נוספות של הגופן, כגון - הדגשה, צבע, וקו תחתון. שנית - שים לב להבדלי תחביר קטנים בין מאפיין Font של פקד Label ובין מאפיין Font של פקד CommonDialog. מאפיין Font של פקד Label (כמו גם של אובייקטים אחרים כגון פקד TextBox) הוא אובייקט כשלעצמו (אובייקט גופן), ותיבת הדו-שיח - CommonDialog מאחסנת כל תכונה של הגופן (שם, גודל, הדגשה, וכיוצא בזה) תחת מאפיין נפרד. בתיבת הטקסט מוצגים בנפרד גם המאפיינים הישנים לצורך השוואה, אך מומלץ כי תתעלם מהם. לדוגמה, אם ברצונך להגדיר גופן עבור שתי תיבות טקסט, תוכל להגדיר תיבה אחת ישירות מתוך מאפייני הפקד CommonDialog ותיבה שנייה פשוט על ידי השוואתה לתיבה הראשונה:

```
set txtAddress.Font = txtName.Font
```

בתרשים 7.7 מופיעה תיבת הדו-שיח Font, כפי שהיא מוצגת למשתמש. התיבה המסומנת שבתרשים מכילה גופני מסך וגופני מדפסת גם יחד.



תרשים 7.7: הינך יכול להתאים את תיבת הדו-שיח Font כך שתציג גופני מסך, גופני מדפסת או את שניהם גם יחד

טבלה 7.4 מציגה את מאפייני הפקד האפשריים אל מול תכונות הגופן הנשלטות באמצעותם.

טבלה 7.4: מאפיינים של פקד CommonDialog, המכילים תכונות גופן

מאפיין	תכונת גופן
FontName	שם הגופן הבסיסי
FontSize	גובה הגופן בנקודות
FontBold	האם נבחרה הדגשה
FontItalic	האם נבחר מצב הטיה
FontUnderline	האם נבחר קו תחתון

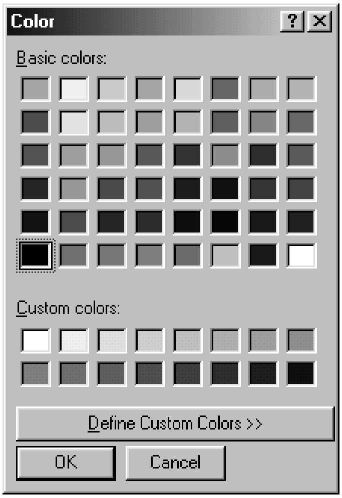
המידע שבטבלה יכול לשמש להגדרת הגופן עבור כל אובייקט בתוכניתך, או אפילו עבור אובייקט Printer. זכור להשתמש במערכת העזרה לצורך קבלת רשימה מפורשת של קבועי הדגלונים האפשריים.

הערה:

כדי לראות את רשימת הקבועים בעזרה, חפש מאמר בשם CommonDialog Control Constants.

תיבת הדו-שיח Color (צבע)

תיבת הדו-שיח Color של הפקד CommonDialog, מאפשרת למשתמש לבחור צבעי רקע קדמי או אחורי עבור טפסים ופקדים (ראה תרשים 7.8). המשתמש יכול לבחור אחד מן הצבעים השגרתיים, או לחילופין - ליצור צבעים מותאמים אישית.



תרשים 7.8: תיבת הדו-שיח Color מאפשרת למשתמש לבחור צבע באופן גרפי ומשלבת בתוכנית את הצבע הנבחר כערך הקסה-דצימלי

תהליך שיבוצן של הגדרות הצבע בפקד CommonDialog, דומה בבסיסו לשיבוצן של הגדרות הגופנים: הגדר את מאפיין Flag כך שיכיל את הקבוע cdlCCRGBInit, והפעל את השיטה ShowColor של הפקד.

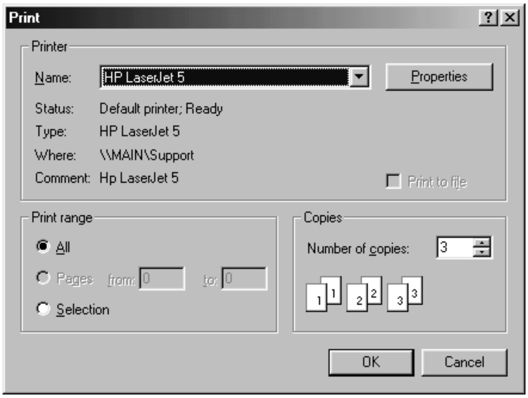
כאשר המשתמש בוחר צבע מתיבת הדו-שיח, ערך הקסה-דצימלי, המייצג את הצבע הנבחר, מאוחסן במאפיין Color של הפקד. כדי לבחון את פעולתה של תיבת הדו-שיח Color, החלף את הקוד בשגרת האירוע cmdTest_Click (בפרויקט הדוגמה) בשורות הקוד הבאות:

```
cdlTest.Flags = cdlCCRGBInit
cdlTest.ShowColor
Form1.BackColor = cdlTest.Color
```

כמו בתיבות הדו-שיח האחרות, תוכל לשנות את מאפייני תיבת Color על ידי שימוש במאפיין Flags. לדוגמה, אם ברצונך לאפשר שימוש בצבעי ברירת מחדל בלבד, השתמש בדגלון cdlCCPreventFullOpen אשר גורם לנטרול האפשרות Define Custom Colors (הגדר צבעים מותאמים אישית), והופך אותה לבלתי נגישה עבור המשתמש. לצורך קבלת רשימה מפורטת של הדגלונים האפשריים עבור תיבת Color והשפעותיהם, פנה אל הנושא Color Dialog\Flags Property (Color Dialog) במערכת העזרה של Visual Basic.

תיבת הדו-שיח Print (הדפסה)

תיבת דו-שיח נוספת אשר ניתן להציג באמצעות הפקד CommonDialog היא התיבה Print (הדפסה). תיבה זו מופיעה בדרך כלל, בסמוך לביצוע עבודת הדפסה ומאפשרת למשתמש לבחור את המדפסת אשר תבצע את העבודה ואת מאפייני ההדפסה (ראה תרשים 7.9). תיבת הדו-שיח כוללת בין היתר, אפשרות להגדיר אילו עמודים יודפסו, בכמה עותקים, וכן אפשרות להדפסת קובץ.



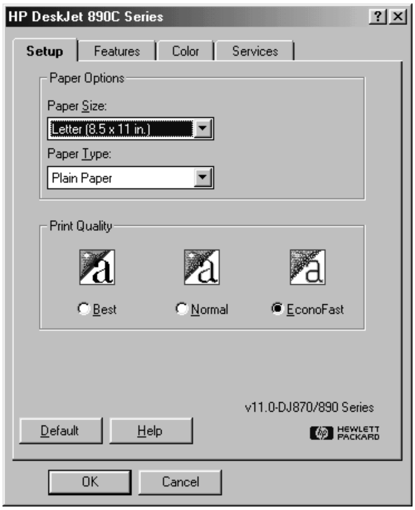
תרשים 7.9: תיבת הדו-שיח Print היא אמצעי עקבי ונוח למשתמשי התוכנית, להגדרת מאפייני ההדפסה

כדי להפעיל את תיבת הדו-שיח Print, קרא לשיטה ShowPrinter של הפקד CommonDialog. אין צורך להגדיר דגלונים לפני כן.

כאשר תיבת הדו-שיח Print מופיעה על המסך, המשתמש יכול לבחור מדפסת רצויה מתוך רשימת המדפסות (Name), הנמצאת בחלק העליון. ברשימה מוצגות כל המדפסות המותקנות במערכת של המשתמש. מתחת לרשימת המדפסות נמצא שדה המצב (Status Line), המציין את מצבה הנוכחי של המדפסת (פנויה או תפוסה).

אם המשתמש מעוניין לשנות מאפייני מדפסת מסוימת (לדוגמה - גודל הנייר והשוליים), עליו לבחור אותה מתוך רשימת המדפסות וללחוץ על לחצן Properties (מאפיינים), הצמוד לה. פעולה זו גורמת להופעת גיליון המאפיינים של המדפסת שנבחרה, ראה תרשים 7.10. המאפשר למשתמש לשלוט בכל מאפייני המדפסת, בדומה לאפשרויות השליטה מלוח הבקרה של Windows.

המידע שנקלט מן המשתמש מוכנס לתוך מאפייניה של תיבת הדו-שיח Print. המאפיינים FromPage ו-ToPage מצביעים על העמודים בהם בחר המשתמש להתחיל את ההדפסה ולסיימה. מאפיין Copies מצביע על מספר העותקים בהם תתקבל ההדפסה. נתונים אלה הם מיקדמיים בלבד, תיבת הדו-שיח אינה מבצעת את עבודת ההדפסה באופן אוטומטי.



תרשים 7.10: תיבת הדו-שיח Properties, המופיעה על המסך בעת לחיצה על לחצן Properties, מאפשרת לך לקבוע את גודל הנייר, רוחב השוליים, ותכונות מדפסת נוספות

לבחינת פעולתה של תיבת הדו-שיח Print, החלף את הקוד בשגרת האירוע cmdTest_Click בשורות הקוד הבאות:

```
cdlTest.Flags = cdlPDDisablePrintToFile
cdlTest.Copies = 3
cdlTest.PrinterDefault = True
cdlTest.ShowPrinter
MsgBox "The default printer is: " & Printer.DeviceName
```

הקוד שבדוגמה עושה שימוש במאפיינים ודגלונים לנטרול האפשרות Print To File והגדרת שלושה עותקים כמצב ברירת מחדל. בנוסף, המאפיין PrinterDefault מצביע על כך שתיבת הדו-שיח תשתמש בהגדרות הפקד לצורך התאמת מאפייני מדפסת ברירת המחדל של המערכת. שימושים נוספים שניתן לעשות באמצעות תיבת הדו-שיח Print, תוכל למצוא במערכת העזרה של Visual Basic.

תיבת הדו-שיח Help (עזרה)

אופן שימוש זה בפקד CommonDialog גורם להפעלת מנוע העזרה של Windows על ידי קובץ WINHELP32.EXE (אם מותקנים במחשב גרסאות Internet Explorer 4 ומעלה, או Windows 98, שם קובץ מנוע העזרה הוא HH.exe). כדי להשתמש בתיבת Help, עליך לבצע מספר פעולות: הגדר במאפיין HelpFile של הפקד, שם ומיקום עבור קובץ עזרה מתאים בעל סיומת HLP. בנוסף, הגדר במאפיין HelpCommand את סוג העזרה שתוצע למשתמש על ידי מנוע החיפוש. בתום הגדרת המאפיינים, השתמש בשיטה ShowHelp להפעלת מערכת העזרה. עתה, יכול המשתמש לנווט במערכת העזרה תוך שימוש בקובץ העזרה של תוכניתך.

יצירת תיבות דו-שיח משלך

פקד CommonDialog מאפשר לשלב בתוכנית מיגוון סוגים של תיבות דו-שיח. אך קיימות משימות מיוחדות אשר אינן ניתנות לביצוע באמצעות התיבות הרגילות. לדוגמה, אם ברצונך להתאים כותרות משלך ללחצנים בתיבה, אינך יכול להשתמש בתיבת הודעה רגילה. כמו כן, תיבת הודעה רגילה מכילה לכל היותר שלושה לחצנים (להוציא את לחצן Help), עובדה היכולה להוות מגבלה במצבים מסוימים. חשוב על תיבת הדו-שיח Input. לא ניתן לשלב בה סמלים והיא אינה יכולה לטפל ביותר מפריט קלט אחד. לפיכך, במצבים בהם פקד CommonDialog אינו מתאים, תוכל להפוך את הטופס הסטנדרטי של Visual Basic לתיבת דו-שיח מותאמת אישית.

יצירת תיבת דו-שיח מותאמת אישית

כל שעליך לעשות הוא למלא אחר ההנחיות הבאות:

1. היכנס למאפיין **BorderStyle** של הטופס, ובחר באפשרות **3 - Fixed Dialog** (תיבת דו-שיח קבועה). תיבות דו-שיח מתאפיינות בכך שלא ניתן לשנות את גודלן. פעולה זו גורמת גם להסרתם של הלחצנים הגדל/שחזר והקטן.
2. הסר את הסמל של הטופס על ידי מחיקת הגדרות המאפיין **Icon**.
3. במאפיין **StartUpPosition** של הטופס בחר באפשרות **1 - CenterOwner**, כך שתיבת הדו-שיח תופיע תמיד במרכז היישום המארח.
4. הוסף פקדים על פי הצורך כדי לקבל את המידע הדרוש מן המשתמש. השתמש בפקדי **Label** ככותרות לתיבות הדו-שיח.
5. השתמש בפקדי **Frame** (מסגרת) לתיחום פקדים אחרים בעת הצורך (כדוגמה, ראה את פקד המסגרת Print Range Area בתיבת הדו-שיח Print).
6. הוסף לחצן **OK**, לצורך אישור בחירת המשתמש.
7. הוסף לחצן **Cancel** המאפשר למשתמש לבטל את המהלך.
8. בעת הצגת תיבת הדו-שיח, דאג שתופיע באופן מודאלי (FrmDialog.Show vbModal).

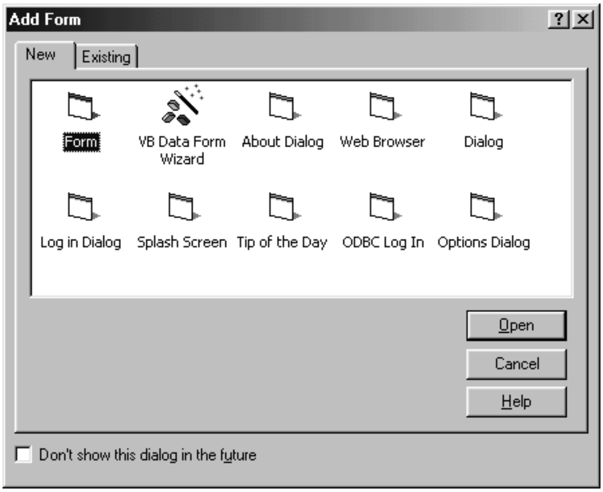
הרגש חופשי להסתייע בתיבות הדו-שיח הנפוצות של Windows, או בתיבות מותאמות אישית המופיעות ביישומים אחרים כדי לשאוב רעיונות, באמצעותם תוכל לעצב תיבות דו-שיח משלך.

שימוש בתבניות טופס עבור תיבות דו-שיח אחרות

חברת Microsoft שילבה ב- Visual Basic מספר **תבניות טופס** (Form Templates) המייצגות תיבות דו-שיח מותאמות אישית שיכולות להיות שימושיות עבורך. תוכל להוסיף בקלות לפרויקט שלך את אחת מתבניות הטופס של Visual Basic על ידי בחירת הסוג המתאים בעת הוספת טופס חדש. להלן פירוט תיבות הדו-שיח האפשריות:

- תיבת **About**. מאפשרת הצגת מידע למשתמש בנוגע לתוכניתך. (תיבה זו מצויה בכל אחת מתוכנות Microsoft בתפריט **עזרה**, **אודות**).
- תיבת **Log In**. מאפשרת למשתמש להכניס סימן מזהה וסיסמה.
- תיבת **Options**. דומה לתיבות הדו-שיח Property Pages ו-Options של Visual Basic.
- תיבת **Tip**. מאפשרת לשלב פונקציה של "העצה היומית" בתוכניתך.

כדי לשלב בתוכניתך אחת מן התיבות שברשימה, בחר **Project**, **Add Form** או לחץ על לחצן **Add Form**. פעולה זו תגרום להופעתה של תיבת הדו-שיח **Add Form**, המוצגת בתרשים 7.11. כעת, בחר אחת מן התבניות המוצעות והתאם אותה לצרכי התוכנית שלך.



תרשים 7.11: מספר תיבות דו-שיח מוצעות לך בצורה של תבניות טופס

מכאן...

בפרק זה למדת כיצד השימוש בתיבות דו-שיח יכול לסייע למשתמש בבחירת קבצים, מדפסות, וגופנים עבור התוכנית. כמו כן למדת כיצד מסייעות תיבת ההודעה (MsgBox) ותיבת הקלט (InputBox) למסירת מידע למשתמש, ולקבלת החלטות ופיסות מידע בודדות.

- בפרק 3 - **אבני היסוד של Visual Basic**, תוכל למצוא מידע נוסף בנושא עיצוב הטפסים ושילובם בתוכניותיך.
- פרק 5 - **תגובה באמצעות שגרות אירוע**, יעניק לך היכרות עם כתיבת הקוד המאפשר לטפסים להגיב לפעולות המשתמש.
- בפרק 9 - **יסודות התכנות ב- Visual Basic**, תמצא דיון מעמיק יותר בנושא כתיבת הקוד ב- Visual Basic.

8

שימוש במשתנים לאחסון מידע

מה בפרק?

- היכרות עם משתנים
- הצהרה על משתנים
- מערכי משתנים
- היכן כדאי לעשות שימוש במשתנים
- שימוש בפקודה Option Explicit
- במה שונים הקבועים

כדי שתוכניות Visual Basic תהיינה שימושיות באמת, עליהן לדעת כיצד לשמור מידע באופן זמני. Visual Basic, כמו כל שפת תכנות אחרת, צריכה לזכור מידע במהלך פעולת התוכנית. יכולת זו מתאפשרת על ידי שימוש במשתנים ובקבועים.

פרק זה יבחן את סוגי המשתנים והקבועים הזמינים למפתחי Visual Basic, ויראה כיצד השימוש בהם מאפשר את שמירת המידע הנחוץ.

היכרות עם משתנים

באופן פשטני, משתנים נועדו כדי לשמור מידע בזיכרון המחשב במהלך פעולת התוכנית. משתנה מוגדר באמצעות שלושה מרכיבים:

- שם המשתנה (אשר מגדיר את מיקומו בזיכרון).
- סוג המידע הנשמר.
- המידע עצמו.

נניח שהוטלה עליך המשימה הבאה: "ספור את כל המכוניות שבמגרש החניה". בזמן ספירת המכוניות, אתה שומר מידע בתוך משתנה. מיקום המשתנה הזה בזיכרון מקביל לפנקס או לראש שלך. סוג המידע שנשמר הוא מספר, והמידע הנשמר הוא מספר המכוניות הנוכחי.

כפי שהשם **משתנה** (Variable) מרמז, המידע שנשמר במשתנה יכול להשתנות (להתחלף) מפעם לפעם. בדוגמה שלנו, כמות המכוניות שאתה סופר גדלה מעת לעת. יש שתי פעולות בסיסיות שניתן לבצע עם כל משתנה: שמירת מידע (כתיבה על הפנקס) והשבת מידע (קריאת הכתוב על הפנקס).

מתן שמות למשתנים

כל משתנה חייב להיות מזוהה בשם, כדי שניתן יהיה לתת לו ערכים. ייתכן שכבר נתקלת בדוגמאות קודמות בהוראות מעין זו:

```
Dim X As Integer
```

דוגמה זו ממחישה את הוראת Dim קיצור של **Dimension**, המשמשת לצורך הצהרה על משתנים (עיין בסעיף **הצהרה על משתנים**, המופיע בשלב מאוחר יותר בפרק). בעת הגדרת משתנה, אנו למעשה אומרים ל- Visual Basic: "שרייני מקום בזיכרון לצורך אחסון מידע משתנה; קראי לו X". שתי המילים האחרונות בפקודת Dim אומרות ל- Visual Basic מה סוג המידע שבכוונתך לשמור, במקרה זה - מספרים שלמים. מידע זה מסייע לתוכנה לדעת מה כמות הזיכרון שיש לשריין עבור המשתנה שהוגדר.

קיימת גמישות רבה מאוד בנוגע להקצאת שמות משתנים. שם המשתנה יכול להיות קצר ופשוט, או להיות תיאור מפורט של סוג המידע אותו הוא מכיל. בדוגמה לעיל X הוא אומנם שם משתנה חוקי לגמרי, אך אין הוא מסביר לקורא את משמעות המידע השמור בו.

למרות הגמישות הרבה, קיימים עדיין מספר חוקים להם עליך לציית בעת מתן שמות למשתנים:

- שם המשתנה חייב להתחיל באות. לא במספר או בתו אחר.
- שארית השם יכולה לכלול אותיות, מספרים ו/או סימני קו תחתון. מרווחים, נקודות וסימני פיסוק אחרים אינם מותרים.
- השם חייב להיות ייחודי ב**טווח ההכרה** (Scope) שבו נמצא המשתנה (טווח ההכרה מתייחס להקשר של המשתנה, כפי שנראה בהמשך).
- אורך השם לא יעלה על 255 אותיות.
- השם לא יהיה אחת מהמילים השמורות של Visual Basic.

מומלץ לתת למשתנים שמות אשר יתארו את תפקידם, זאת כדי להפוך את הקוד נוח יותר לקריאה. מאידך, יש להקפיד על מתן שמות קצרים ככל האפשר, כדי להקל על כתיבת הקוד. מפתחים רבים מצרפים לשם המשתנה **קידומת** (prefix) המצביעה על סוג המידע השמור; קידומות אלו הן, בדרך כלל, וריאציות שונות של **שיטת מתן השמות ההונגרית** (Hungarian Naming Convention), ומבוססות על אות או שתי אותיות קטנות בתחילת שם המשתנה, ואחריהן אות גדולה.

לדוגמה התחילית s בשם sCity, מצביעה על כך שמדובר במשתנה המכיל ערך **מחרוזת** (String). טבלה 8.1 מציגה מספר קידומות נפוצות בהן מומלץ להשתמש בעת מתן שמות למשתנים. שיטה זו (המתוארת ב- Microsoft Knowledge Base במאמר Q110264, **Microsoft Consulting Services Naming Conventions for VB**) באה לידי ביטוי ברבות מן הדוגמאות המובאות בספר זה.

טבלה 8.1: קידומות לשמות המשתנים

סוג משתנה	קידומת	דוגמה
String	s	sFirstName
Integer	n	nAge
Long Integer	l	lPopulation
Single (Floating Point)	f	fAverageGrade
Double	d	dThrustRatio
Currency	c	cPayRate
Boolean	b	bTaxable

שמות משתנים המכילים מספרים **שלמים** (Integers) מתחילים בקידומת n ולא בקידומת i (אשר נראית הגיונית יותר), זאת משום שהקידומת i שמורה לייצוג אינדקסים.

סוגי משתנים

עד כה למדת מהו תפקיד המשתנה וכיצד לתת לו שם. אך מה סוג המידע שניתן לשמור במשתנה? התשובה הפשוטה היא **כמעט כל דבר**. משתנה יכול להכיל מספר; מחרוזת טקסט; או הפניה לאובייקט כמו טופס, פקד, או בסיס נתונים. פרק זה מתרכז בשימוש במשתנים לצורך אחסון מספרים, מחרוזות וערכים לוגיים. אופן השימוש במשתנים לייצוג אובייקטים ומסדי נתונים מכוסה מאוחר יותר בפרק 16 **מחלקות: שימוש חוזר ברכיבים**, ופרק 26 **שימוש באובייקטי גישה לנתונים (DAO)**.

לכל סוג משתנה יש דרישות זיכרון משלו, והוא מתוכנן לעבוד ביעילות עם סוגים שונים של מידע. לפיכך, אין זה אפשרי לאחסן מחרוזת כמו "Hello" בתוך משתנה המוגדר כ-Integer (מספר שלם).

טבלה 8.2 מציגה חלק מסוגי המשתנים הסטנדרטיים הקיימים ב- Visual Basic. בנוסף, מראה הטבלה את טווח הערכים אשר משתנה יכול להכיל, ואת כמות הזיכרון הנדרשת. כאשר הדבר אפשרי, מומלץ להשתמש במשתנים בעלי דרישות זיכרון נמוכות (שבשפה מקצועית נקראים משתנים ״זולים״), כדי לחסוך במשאבי מערכת.

טבלה 8.2: משתנים מסוגלים לשמור סוגים רבים של מידע

סוג	המידע הנשמר	צריכת זיכרון	טווח ערכים
Integer	מספרים שלמים	2 Byte	-32,768 עד +32,767
Long	מספרים שלמים	4 Byte	(בערך) -/+ 2 מיליארד
Single	מספרים עשרוניים	4 Byte	1E-45 -/+ עד 3E38
Double	מספרים עשרוניים	8 Byte	5E-324 -/+ עד 1.8E308
Currency	מספרים עד 15 ספרות משמאל לנקודה ועד 4 ספרות מימין לנקודה	8 Byte	9E14 -/+
String	מידע טקסטואלי	Byte אחד לכל תו	עד 65,400 תווים למחרוזת בגודל קבוע, ועד 2 מיליארד תווים למחרוזת דינמית
Byte	מספרים שלמים	1 Byte	0 עד 255
Boolean	ערכים לוגיים	1 Byte	True או False
Date	מידע על תאריך ושעה	8 Byte	1/1/100 עד 12/31/9999
Object	התהוות מחלקות; אובייקטי OLE	4 Byte	אין
Variant	כל אחד מסוגי המידע שלעיל	16 Byte + 1 Byte לכל תו	אין

בנוסף לסוגי המשתנים שבטבלה, ניתן ליצור גם **סוגים המוגדרים על ידי המשתמש** (User Defined Types). שורות הקוד הבאות מדגימות שימוש במשתנים מסוג זה:

```
' הגדרת סוג משתנה חדש
Private Type Point
            X As Integer
            Y As Integer
End Type

' הגדרת משתנה מהסוג החדש
Private Sub Command1_Click()
            Dim MyPoint As Point
            MyPoint.X = 3
            MyPoint.Y = 5
End Sub
```

כפי שניתן לראות בדוגמה, הגדרת סוג משתנה חדש מתבצעת באמצעות הפקודה Type. שים לב כי הגדרת סוג המשתנים חייבת להתבצע באזור ההצהרות הכלליות. במקרה זה הגדרנו מתוך הטופס סוג פרטי הנקרא Point. כדי ליצור משתנה מסוג חדש, עליך להשתמש בפקודה Dim בדומה להצהרה על משתנים מסוגים אחרים. ניתן לגשת לחלקים במשתנה באמצעות סימן הנקודה (.).

כפי שכבר ציינו, קיימים מספר סוגי משתנים המיועדים לטיפול במסדי נתונים בלבד (כגון סוגי המשתנים Database, File ו-Recordset). מידע הקשור לסוגי משתנים אלה מועבר ל- Visual Basic בעת הוספת הפניה ל**ספריית סוגים** (Type Library). ספריית סוגים היא קובץ **DLL** (ספריית קישור דינמית), או קובץ היוצר בתוכניתך זמינות למשתנים המטפלים בבסיסי נתונים. דוגמה טובה לשימוש בספריית סוגי משתנים תמצא בפרק 22, **שימוש ב-OLE לשליטה על יישומים אחרים**.

ראה: סעיף **ספריית האובייקטים של Word**, בפרק 22.

הצהרה על משתנים

קודם לכן בסעיף **מתן שמות למשתנים**, ראית דוגמה לפקודה Dim, באמצעותה הגדרת את שם וסוג המשתנה שלך. אולם Visual Basic אינה מחייבת להצהיר במפורש על משתנה טרם השימוש בו. אם המשתנה לא הוגדר בצורה מסודרת, Visual Basic תיצור אותו בעצמה כמשתנה מסוג ברירת המחדל, בדרך כלל Variant. משתנה המוגדר כך מסוגל להכיל מידע מכל סוג, אך יש לו שני חסרונות: הוא עלול לבזבז שטח זיכרון יקר, וייתכן כי לא יתאים לעבודה עם פונקציות המצפות למשתנה מסוג ספציפי.

הצהרת המשתנים מראש לפני שימוש בהם הינה מנהג תכנותי טוב. לכן, כדאי שתבחן את שתי הדרכים להצהרה על משתנים - הצהרות **מפורשות** (Explicit) והצהרות **מרומזות** (Implicit). שים לב גם למקרה המיוחד של **מחרוזות בעלות גודל קבוע**.

הצהרה מפורשת

הצהרה מפורשת (Explicit Declaration) - מונח זה משמעותו שימוש בפקודות קוד לשם הגדרת שמות וסוגי המשתנים בתוכנית. הפקודות אינן משמשות להצבת ערכים במשתנים, אלא רק מגדירות ל- Visual Basic מה שמות המשתנים שישולבו בתוכנית, ומה סוג הנתונים שיכילו.

באפשרותך להשתמש בכל אחד מהמשפטים הבאים כדי להגדיר מפורשת סוג משתנה:

```
Dim varname [As vartype][, varname2 [As vartype2]]

Private varname [As vartype][, varname2 [As vartype2]]

Static varname [As vartype][, varname2 [As vartype2]]

Public varname [As vartype][, varname2 [As vartype2]]
```

Dim, Private ו-Public הן מילות מפתח ב- Visual Basic אשר מגדירות באיזה אופן תשתמש התוכנית במשתנה המוגדר (בסעיפים הבאים תוכל ללמוד יותר היכן וכיצד לעשות שימוש במילות מפתח אלו). המילים *varname* ו-*varname2* מייצגות את שמות המשתנים אשר אתה מעוניין להגדיר. ניתן להגדיר מספר משתנים בפקודה אחת, כל עוד שמותיהם מופרדים באמצעות פסיקים. כל שם משתנה חייב הגדרה נפרדת לסוג המשתנה, אם לא יוגדר סוג המשתנה עבור כל שם משתנה בנפרד, המשתנה ייחשב כ-Variant. בדוגמה לעיל, מופיעים אומנם שני משתנים בלבד, אך ניתן לציין מספר רב יותר של שמות. למעשה, בשורה אחת של חלון הקוד ניתן להקליד מעל אלף תווים, אם כי מעשית, מומלץ להימנע מכתיבת שורות החורגות מרוחב החלון המוצג על המסך. הקפדה על כתיבת שורות באורך סביר, מקלה על קריאות הקוד שלך וחוסכת פעולות מיותרות כמו גלילת החלון ימינה ושמאלה בזמן קריאתו.

שורות הקוד הבאות מדגימות כיצד להשתמש בפקודות לשם הצהרה על משתנים:

```
Private nNumVal As Integer
Private nAvgVal as Integer, vInputVal as Variant
Static fClcAverage as String
Dim sFirstName as String
```

הצהרה מרומזת

לרוב רצוי להגדיר את המשתנים בעזרת פקודת Dim או אחת מהפקודות האחרות המשמשות לצורך הצהרה מפורשת. אולם, במקרים רבים ניתן גם להגדיר את סוג המשתנה באמצעות **הצהרה מרומזת** (Implicit Declaration). בהצהרה מסוג זה, יש לסיים את שם המשתנה בתו מיוחד, בעת הצבת ערך בפעם הראשונה. בטבלה הבאה מוצגים התווים אותם יש להוסיף לכל אחד מסוגי המשתנים הקיימים.

טבלה 8.3: תווים מיוחדים לסוגי המשתנים

סוג המשתנה	תו
Integer	%
Long	&
Single	!
Double	#
Currency	@
String	$
Byte	אין
Boolean	אין
Date	אין
Object	אין
Variant	אין

כאשר Visual Basic נתקלת בפעם הראשונה בתוכנית במשתנה שהוגדר באופן מרומז, היא מקצה לו אוטומטית מקום בזיכרון. הצהרה מרומזת על המשתנים שמהדוגמה הקודמת נעשית בצורה הבאה:

```
nNumVal% = 0                 ' Integer-מוגדר בזיכרון כ
nAvgVal% = 1                 ' Integer-מוגדר בזיכרון כ
vInputVal = 5                ' Variant-מוגדר בזיכרון כ
fClcAverage! = 10.1          ' Single-מוגדר בזיכרון כ
sFirstName$ = "Lauren"       ' String-מוגדר בזיכרון כ
```

שים לב כי המשתנה vInputVal אינו מכיל את תו סוג המשתנה, ולכן Visual Basic תתייחס אליו כמשתנה מסוג Variant.

מחרוזות בעלות גודל קבוע

רוב המחרוזות בהן נעשה שימוש בתוכניות Visual Basic הן **מחרוזות בעלות גודל משתנה** (Variable Length Strings). מחרוזות אלו יכולות להכיל כל כמות של טקסט, עד לשני מיליארד תווים בערך. בזמן שמידע נשמר בתוך המשתנה, גודלו מותאם כך שיוכל להכיל את המחרוזת. ההצהרות המפורשות והמרומזות עליהן למדת זה עתה, תיצורנה מחרוזות בעלות גודל משתנה, אולם ב- Visual Basic קיים סוג נוסף של מחרוזות: **מחרוזות בעלות גודל קבוע** (Fixed Length Strings).

מחרוזת בעלת גודל קבוע, כפי ששמה מרמז, שומרת על גודלה, ללא תלות במידע המאוחסן בה. אם במחרוזת כזו מוצב ביטוי קצר יותר מאורכה הקבוע, אזי שארית המחרוזת תמולא במרווחים. אם הביטוי ארוך יותר מהגודל הקבוע, אזי יקוצצו התווים העודפים.

ניתן להצהיר על מחרוזת בעלת גודל קבוע באמצעות הצהרה מפורשת בלבד:

```
Dim varname As String * strlength
```

שים לב כי הצהרה זו שונה מעט מן ההצהרה הקודמת של משתנה המחרוזת. הצהרת מחרוזת בעלת גודל קבוע מכילה כוכבית (*) המציינת כי מדובר במחרוזת בעלת גודל קבוע. הפרמטר האחרון, *strlength*, מגדיר ל- Visual Basic את גודלה הקבוע של המחרוזת (כלומר מספר התווים המקסימלי שביכולה להכיל).

מחרוזת בעלת אורך קבוע יכולה לשמש למשל לבניית קובץ טקסט המופרד באמצעות פסיקים. נניח ש-25 תווים מהקובץ משמשים לביטוי שמו של אדם. הקוד הבא יקצץ אוטומטית את כל התווים במחרוזת מעבר לתו 25:

```
dim sName As String * 25
sName = "This text is too long for the string variable."
```

הצבת ערך שאורכו קצר מ-25 תווים ב-sName תגרום ששאר התווים במשתנה זה ימולאו ברווחים.

הערה:

אחד השימושים הנפוצים של מחרוזות בעלות אורך קבוע הוא בהשבת מידע מפונקציית API של חלונות, נושא זה נדון בפרק 20 **גישה ל- API של Windows**.

מערכי משתנים

המשתנים שנדונו עד כה היו כולם משתנים יחידים. אולם, יש מקרים רבים בהם יעיל יותר לעבוד עם **מערכים** (Arrays). מערך הוא קבוצת משתנים מאותו סוג, המוגדרים תחת שם זהה. בדרך זו קל יותר לעבד קבוצות של שדות מקורבים. לדוגמה, נניח כי אתה נעזר בקבוצת משתנים המנהלים מעקב אחר נפח המכירות בכל אחד מארבעת האזורים של החברה שלך. אתה יכול להגדיר משתנה מטבע לכל אזור, ועוד משתנה כדי להחזיק את סכום המכירות בכל האזורים ביחד, כמו בדרך זו:

```
Dim cRegSales1 As Currency, cRegSales2 As Currency
Dim cRegSales3 As Currency, cRegSales4 As Currency
Dim cTotalSales As Currency
```

כעת, כדי לחשב את סכום המכירות בכל האזורים, תצטרך להשתמש בקוד הבא:

```
cTotalSales = cRegSales1 + cRegSales2 + cRegSales3 + cRegSales4
```

גישה זו אינה מסורבלת במיוחד, אך מה יקרה אם יהיו 20 אזורים, או כמה מאות? קל לראות שעבודה עם מספר גדול של משתנים מקורבים יכולה לגרום לתסבוכת.

ניתן לפשט בקלות דוגמה זו על ידי שימוש במערך. ניצור מערך בשם cRegSales. מערך זה יכול להכיל מספר אלמנטים (מופעי משתנים) הזהה למספר האזורים שלך. ניתן לכתוב מחדש את הדוגמה כך שתתאים לטיפול ב-20 אזורים:

```
Dim cRegSales(1 to 20) As Currency
Dim cTotalSales As Currency
Dim nCounter As Integer
dim sTemp as Integer

        cTotalSales = 0
        For nCounter = 1 To 20
                cTotalSales = cTotalSales + cRegSales(nCounter)
        Next nCounter
        sTemp = "Total sales for all regions = "
        sTemp = sTemp & Format(cTotalSales, "currency")
MsgBox sTemp, vbInformation, "Sales Analysis"
```

שים לב בדוגמה לשימוש ב**לולאה** (Loop). חלק הקוד שמתחיל בהוראה For ומסתיים בהוראה Next מגדיר קבוצת פקודות בתוכנית שתחזורנה על עצמן מספר פעמים (20 במקרה זה). שימוש בלולאות מקצר את עבודת עיבוד המערכים.

ככל שתתקדם בספר זה תיתקל במקרים רבים בהם השימוש במערכים יכול לפשט מאוד את קוד התוכנית שלך.

היכן כדאי לעשות שימוש במשתנים

בנוסף לסוג הנתונים הנשמרים במשתנה, קובעת ההצהרה גם את **טווח ההכרה** (Scope) במשתנה. ניתן לדמות את טווח ההכרה במשתנים לאזורי הכיסוי של טלפון סלולרי. בזמן רכישת טלפון סלולרי, ביכולתך לקבוע עבורו כיסוי ארצי או בינלאומי. מידע זה נצרב לתוך המכשיר. אם תצא מטווח הכיסוי שלך, הטלפון לא יעבוד. בצורה דומה, ניתן להצהיר על משתנים כך שיוכלו לפעול בשיגרה אחת בלבד בטופס, בכל השגרות בטופס, או בכל חלקי התוכנית.

ברירת המחדל של Visual Basic מתייחסת למשתנים שהוגדרו בצורה מרומזת, כמקומיים לשיגרה בה נוצרו. לפיכך, אם אינך משתמש בהצהרה עבור משתנה כלשהו, Visual Basic תתייחס אליו בהכרח כאל משתנה מקומי. על כן, כדי ליצור משתנה שהיקפו אינו מקומי, הינך חייב להשתמש באחת מפקודות ההצהרה.

הערה:

תחום המשתנה נקבע על ידי סוג ההצהרה, וגם על ידי מיקומה. לדוגמה, למילות המפתח Dim ו-Private יש משמעויות שונות בחלקים השונים של הקוד בטופס.

משתנים הזמינים מכל מקום

במרבית התוכניות (אלא אם יש לך טופס אחד בלבד ללא מודולי קוד) תוכל להבחין כי קיים צורך במשתנים אחדים, אשר ניתן יהיה לגשת אליהם מכל מקום בתוכנית. משתנים אלה נקראים **משתנים ציבוריים** (Public Variables). בשפות אחרות, כולל בגרסאות מוקדמות יותר של Visual Basic, ייתכן שמשתנים מסוג זה כונו בשם **משתנים גלובליים** (Global Variables). (למעשה גם גירסה 6 של Visual Basic מזהה עדיין את מילת המפתח Global). משתנים אלה בדרך כלל יכילו מידע כגון שם המשתמש, או הפניה למסד נתונים אשר נמצא בשימוש לכל אורך התוכנית. הם יכולים גם לשמש בתור דגלים אשר מייצגים תנאים מסוימים בתוכנית.

כדי ליצור משתנה ציבורי, עליך לשלב את פקודת ההצהרה Public באזור ההצהרות הכלליות במודול (קובץ BAS) של התוכנית שלך. השורה הבאה מראה כיצד מצהירה הפקודה Public על משתנה ציבורי, השומר את שם המשתמש :

```
Public sUserName As string
```

מילת המפתח Public הינה בעלת מעמד מיוחד בקוד הטופס או המודול המחלקתי. משתנה המוגדר כ-Public, פועל ממש כמו אחד ממאפייני הטופס או המחלקה, אשר זמינים מכל מקום בתוכנית. Visual Basic מתייחסת למשתני Public כשווי מעמד למאפיינים הקבועים של טפסים ופקדים, ולא כמשתנים ציבוריים רגילים. מאפיינים מסוג Public משמשים להעברת מידע בין טפסים וחלקים אחרים בתוך התוכנית. נניח לדוגמה, כי ההצהרה הקודמת ל-sUserName התקיימה בתוך טופס. שורות הקוד הבאות יכולות היו לשמש כדי לשנות ולאחזר את שם המשתמש מהטופס.

```
frmLogin.sUserName = "CKRAMER"
frmLogin.Show vbModal
MsgBox "The name entered was " & frmLogin.sUserName
```

שורת הקוד הראשונה קובעת את תכולת המשתנה הציבורי (וטוענת את frmLogin אם אין הוא טעון עדיין), ממש כאילו הוא אחד ממאפייני הטופס. שורת הקוד הבאה מציגה את הטופס כ**מודאל** (Modal) (פתיחת הטופס בצורה מודאלית מחייבת את סגירתו לפני שהתוכנית תוכל להמשיך, בדומה לחלון סגירת Windows), כך ששורת הקוד השלישית לא מופעלת עד אשר הטופס מוסתר. אם המשתמש משנה את ערך sUserName מתוך הטופס ולאחר מכן מסתיר את הטופס, השורה השלישית של הקוד מציגה תיבת הודעה עם הערך החדש.

שמירת משתנה כמקומי

במקרים בהם אין צורך לגשת למשתנה מכל מקום בתוכנית, אין טעם להשתמש במילת המפתח Public בהצהרה. במקום זאת, ניתן להשתמש במילות המפתח Dim או Private, אשר אומרות ל- Visual Basic להגדיר את המשתנה בתחום השיגרה או הטופס הנוכחי, בהתאמה. בהצהרות מסוג זה, מיקום פקודת ההצהרה קובע את טווח ההכרה במשתנה. אם המשתנה מוגדר בחלק ההצהרות הכלליות בטופס או במודול, המשתנה יהיה זמין לכל שיגרה באותו טופס או מודול. משתנה מסוג זה מכונה **משתנה ברמת**

הטופס (Form level Variable) או **משתנה ברמת המודול** (Module level Variable). אם המשתנה מוגדר בתוך שיגרה, ניתן יהיה לעשות בו שימוש בתוך שיגרה זו בלבד. משתנה מסוג זה מכונה **משתנה מקומי** (Local Variable).

מדוע כדאי לשמור משתנה בצורה הכי מקומית שאפשר? התשובה מתחלקת לשניים:

- **הרגל תכנותי טוב**. קשה יותר לנפות שגיאות ולתחזק קוד המבוסס על משתנים ציבוריים. לדוגמה, הבט בשני המימושים של השיגרה הפשוטה, CalcArea:

```
Sub CalcArea()
        MsgBox "The Answer is " & nHeight * nWidth
End Sub

Sub CalcArea(nHeight As Integer, nWidth as Integer)
        MsgBox "The Answer is " & nHeight * nWidth
End Sub
```

  אם משימתך תהיה לנפות שגיאות בשיגרה CalcArea, תיווכח שהדוגמה הראשונה קשה ביותר לניפוי. החישוב מותנה בכך ששני משתנים ציבוריים (nHeight ו-nWidth) יהיו מכוונים נכון. בדוגמה השנייה הבעיה אינה כה קשה משום ש-nHeight ו-nWidth מוגדרים כפרמטרים של הפונקציה, כך שהמפתח יכול לקרוא לפונקציה מחלון הפקודות המיידי מבלי להציב לפני כן ערכים במשתנים. כמובן, שכדי לגרום ל-CalcArea להיות מודולרית וניתנת לשימוש חוזר ככל אפשר, עדיף יהיה להפוך אותה לפונקציה.

- **שימוש במשאבים:** הסיבה השנייה לשמירה על מקומיות המשתנים היא עניין של הגיון בריא. כאשר משתנה מוגדר כ-Public, Visual Basic חייבת להפוך אותו זמין לכל הטפסים והמודולים בתוכנית שלך. היקפו המוגבל של משתנה מקומי מצריך כמות קטנה יותר של משאבים, ובכך מייעל את התוכנית.

שימוש במשתנים סטטיים

מרבית המשתנים אשר מוגדרים בתוך שיגרה, "נזרקים" על ידי Visual Basic עם סיום השיגרה. אך יש מקרים בהם תהא מעוניין לשמור את ערך המשתנה גם בסיום פעולת השיגרה. לדוגמה, כאשר יש צורך לקרוא לשיגרה מספר פעמים, וערך משתנה מסוים תלוי, בכל פעם, בערך שניתן לו בפעם הקודמת בה הופעלה השיגרה.

דוגמה: יש לך שיגרה המדפיסה את מספר העמוד בראש כל עמוד בדוח. אותה שיגרה צריכה לשמור את מספר העמוד שהודפס בעת הפעולה הקודמת, כדי שתוכל להגדילו בכל פעם במספר אחד.

כדי ליצור משתנה אשר שומר את ערכו, יש להשתמש במילת המפתח Static בתוך הצהרת המשתנה. מכך מבינה Visual Basic שני דברים: המשתנה הוא מקומי לאותה שיגרה, וכן יש לשמור את ערך המשתנה כי ייתכן שיהיה בו צורך מאוחר יותר. הנה דוגמה לשילוב מילת המפתח Static בהגדרת המשתנה:

```
Static nPageNumber As Integer
```

הערה:

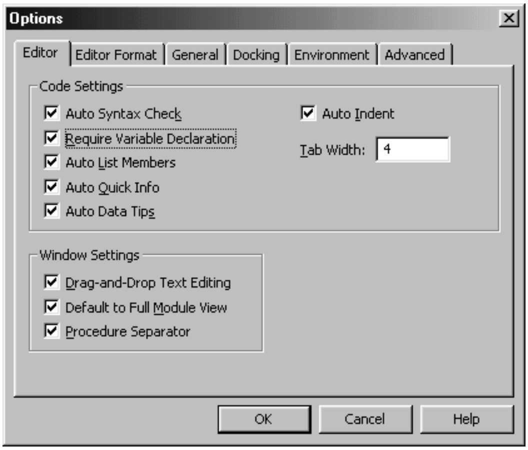
שימוש במילת המפתח Static בראש הפונקציה/השיגרה (כמו ב: **Static sub MySub**), הופכת את המשתנים בשיגרה לסטטיים. השיגרה הבאה מדגימה את העיקרון:

```
Static Sub Test()
        Dim x As Integer
        Debug.Print "Value of x before assignment statement = " & x
                x = 1234
        Debug.Print "Value of x after assignment statement = " & x
End Sub
```

בדוגמה זו, Visual Basic תתייחס למשתנה x בתור משתנה סטטי. בפעם הראשונה שתקרא ל-Test, הפלט יהיה 0 ו-1234. בקריאה הבאה יופק הפלט 1234 עבור שני הערכים - לפני ואחרי.

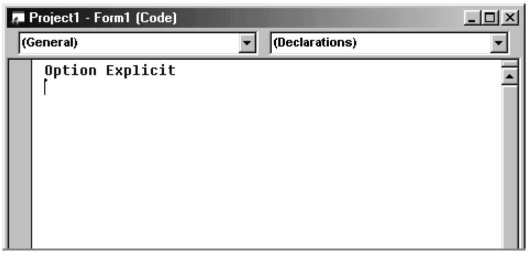
שימוש בפקודה Option Explicit

מוקדם יותר בסעיף **הצהרות משתנים**, למדת כי הצהרה על משתנים, טרם השימוש בהם, הינה הרגל תכנותי טוב. ניתן לגרום ל- Visual Basic "להכריח" אותך להצהיר על המשתנים על ידי שינוי אחת מאפשרויות הסביבה שלה. כדי לעשות זאת, היכנס לתיבת הדו-שיח **Options**, על ידי בחירה ב-**Tools**, **Options**. היכנס לכרטיסיית **Environment** ובחר בתיבת הסימון **Require Variable Declaration** (ראה תרשים 8.1). פעולה זו תגרום ל- Visual Basic לדרוש הצהרה על כל משתנה טרם השימוש בו.



תרשים 8.1: האופציה Require Variable Declaration עוזרת לך להימנע מכתיבה לא נכונה של שמות משתנים

בחירה באפשרות זו תגרום למיקום הפקודה Option Explicit בחלק ההצהרות הכלליות של כל המודולים והטפסים החדשים שתוסיף לפרויקט, ראה תרשים 8.2.



תרשים 8.2: הפקודה Option Explicit מתווספת לתוכניתך

כאשר בטופס או במודול מופיעה הפקודה Option Explicit, אך אתה נמנע מלהצהיר על משתנה, בעת הפעלת התוכנית תתקבל על המסך הודעת השגיאה: Variable not defined (משתנה לא מוגדר). מנפה השגיאות יאיר את המשתנה הפגום ותהליך ההידור ייעצר. הודעה זו מסייעת לך לאתר טעויות שמקורן בשגיאות כתיב. למשל, בעת הצהרה על משתנה באמצעות ההוראה הבאה:

```
Dim sMyName As String
```

אם בהוראה מאוחרת תכתוב לא נכון את שם המשתנה, Visual Basic תגלה את השגיאה עבורך במקום להמשיך ולגרום לתוצאות בלתי צפויות. לדוגמה, אם הוראת Option Explicit מופיעה במודול או בטופס, הפקודה הבאה תגרום להודעת שגיאה; אחרת התוכנית תמשיך לפעול מבלי להציב את הערך במשתנה הרצוי:

```
sMyNme = "Tina Marie"
```

ללא ההוראה Option Explicit, שורת הקוד הקודמת היתה יוצרת משתנה נוסף, במילים אחרות, היו לך שני משתני מחרוזת: sMyName ו-sMyNme.

הערה:

אם תבחר את האופציה Require Variable Declaration לאחר שתתחיל ליצור תוכנית, האופציה לא תשפיע על טפסים או מודולים שנוצרו כבר. במקרה זה, תוכל להוסיף את ההוראה Option Explicit כשורה הראשונה של הקוד באזור ההצהרות הכלליות בכל אחד מהטפסים או המודולים הקיימים.

טיפ:

אם הינך משתמש במעט אותיות גדולות בהצהרות המשתנים (והשגרות), כתוב את הקוד באותיות קטנות. Visual Basic תשנה אוטומטית את גודל האותיות בשמות המשתנים כך שהם ייראו כמו בהצהרה שלהם. דרך זו תספק לך התוויה חזותית מיידית לכך שכתבת נכון את שם המשתנה (גודל האותיות ישתנה רק לאחר הקשת Enter בסוף השורה).

מה שונה בקבועים

שימוש במשתנים הינו רק דרך אחת לשמור מידע בזיכרון המחשב. דרך אחרת היא על ידי שימוש ב**קבועים** (Constants). קבועים מטופלים בדרך מיוחדת. לאחר הגדרתם (אם לא הוגדרו מראש על ידי Visual Basic), אין אפשרות לשנות את ערכם במהלך התוכנית. ניסיון לעשות זאת יגרום להופעת הודעת שגיאה בעת הפעלת התוכנית. נסה לחשוב על קבועים כעל משתנים אשר לא ניתן לשנות את ערכיהם.

כיצד להשתמש בקבועים

קבועים משמשים בדרך כלל להחלפת ערכים אשר קשה לזכור או לכתוב, כגון ערך הצבע של שורת הכותרת בחלון. קל יותר לזכור את הקבוע vbActiveTitleBar מאשר את המספר 2147483646-. ניתן גם להשתמש בקבועים כדי להימנע מכתיבת מחרוזות ארוכות בהן נעשה שימוש במקומות רבים בתוכנית. למשל, ניתן להגדיר קבוע כמו FileErrMsg, אשר יכיל בתוכו את המחרוזות "The requested file was not found".

לעיתים קרובות הם משמשים גם כגורמים המיועדים להמרה, כגון 12 אינצ'ים לרגל, או 3.3 רגל למטר. דוגמת הקוד הבאה מראה כיצד משתמשים בקבועים ומשתנים:

```
Const MetersToFeet = 3.3
nDistMeters = InputBox("Enter a distance in meters")
nDistFeet = nDistMeters * MetersToFeet
MsgBox "The distance in feet is: " & CStr(nDistFeet)
```

שימוש נפוץ נוסף לקבועים הוא צמצום שינויים בקוד, כמו שינוי שם התוכנית, מספר הגירסה וכדומה. ניתן להגדיר קבועים בהתחלת התוכנית ולהשתמש בהם בתוך התוכנית. כאשר מספר הגירסה משתנה, כל שצריך לעשות הוא לשנות את הצהרת הקבוע (במקום לשנות את כל החלקים בתוכנית שמשתמשים במספר הגירסה, למשל). הדוגמה הבאה ממחישה טכניקה זו:

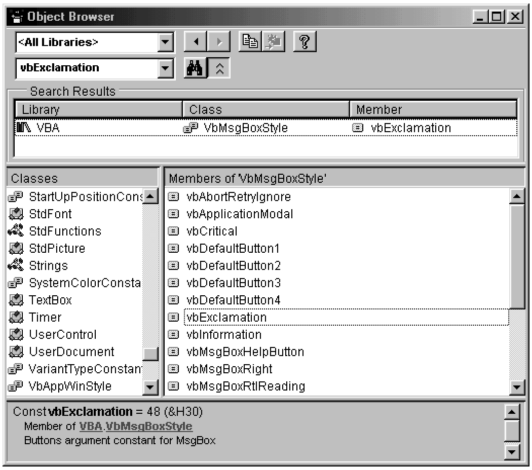
```
Public Const ProgTitle = "My Application Name"
Public Const ProgVersion = "3.1"
```

שים לב לשימוש במילת המפתח Public, אשר גורמת לקבועים אלה להיות זמינים בכל התוכנית (בהנחה שההצהרה הזו נמצאת בתוך מודול).

קבועים המסופקים על ידי Visual Basic

Visual Basic מספקת מספר קבועים מובנים עבור פעולות שונות. קבועים אלה ידועים גם בתור **קבועים פנימיים** (Intrinsic Constants). בין קבועים אלה מצויים קבועי הגדרות צבעים, קבועי גישה לנתונים, קבועי קודים למקלדת, קבועי צורות ועוד. קבועים שימושיים במיוחד הם הקבועים הקשורים לפרמטרים של פקודות, כמו הקבוע vbExclamation אשר נמצא בשימוש בפקודה MsgBox, כפי שכבר למדת בפרקים הקודמים.

הקבועים, אשר בהם ניתן להשתמש ברוב הפונקציות, מתוארים בנושאי העזרה של אותן פונקציות. אם ברצונך לדעת ערך של קבוע מסוים, תוכל להשתמש ב**סורק האובייקטים** (Object Browser), ראה תרשים 8.3. ניתן לגשת אליו בעזרת לחיצה על הסמל שלו בסרגל הכלים, על ידי בחירת **View**, **Object Browser** מתוך התפריטים, או פשוט על ידי הקשה על F2. תוכל להשתמש ברשימה כדי למצוא את הקבוע הרצוי. לאחר בחירתו, יופיעו ערכו ותפקידו באזור הטקסט שבתחתית תיבת הדו-שיח.



תרשים 8.3: סורק האובייקטים מראה את הערך והשימוש של רוב הקבועים הפנימיים ב- Visual Basic

יצירת קבועים משלך

Visual Basic מציעה אומנם קבועים רבים למיגוון מטרות, אך לעיתים תצטרך גם להגדיר קבועים משלך. קבועים מוגדרים באמצעות הפקודה Const המייחסת לקבוע שם וערך, כפי שניתן לראות בתחביר הבא:

[Public | Private] Const c*onstantname* [As *constanttype*] = *value*

אם הוראה זו נראית לך דומה להצהרה על משתנה, אינך טועה. בדומה להצהרה על משתנה, גם בהצהרה על קבוע יש לספק שם עבור הקבוע, וכאופציה - גם את סוג המידע אשר הקבוע יכיל. מילת המפתח Const בתחילת המשפט מיידעת את Visual Basic כי מדובר בשורת קוד המגדירה קבוע. מילת מפתח זו מבדילה בין פקודה זו לבין פקודות אחרות, המציבות ערכים במשתנים. כאשר מגדירים סוג של קבוע, יש להשתמש באותם סוגים המשמשים להגדרת משתנים (רשימת הסוגים נמצאת בטבלה 8.2). לבסוף, כדי להשלים את הגדרת הקבוע, יש להוסיף את סימן השווה (=) ולאחריו את הערך המוצב בקבוע. אם הינך מגדיר קבוע מחרוזת או קבוע תאריך, זכור לתחם את הערך בין סימני מרכאות ("‏) או סולמית (#), בהתאמה.

חשוב לדעת גם על טווח ההכרה בקבוע: החוקים אשר תקפים לגבי קביעת טווח ההכרה במשתנים, אשר נדונו בסעיף **היכן כדאי לעשות שימוש במשתנים**, תקפים גם לגבי קבועים.

מכאן...

פרק זה ערך סקירה כללית בנושא שימוש במשתנים וקבועים ב- Visual Basic.

- בפרק 11 **ניהול הפרויקט: תת-שגרות, פונקציות, וריבוי טפסים**, תלמד את יסודות כתיבת הקוד המהווה את תמצית היישום ב- Visual Basic.

לקבלת מידע נוסף על תכנות, פנה לפרקים הבאים:

- עיין בפרק 9 **יסודות התכנות של Visual Basic**, כדי להעמיק את היכרותך עם השפה.
- עיין בפרק 12 **הפקדים המשותפים של Microsoft**, כדי ללמוד עוד על האירועים הרבים והשונים המיוחסים לפקדים של Visual Basic.

9

יסודות התכנות של Visual Basic

מה בפרק?

- כתיבת משפטים
- שימוש במשפטי השמה
- שימוש בפעולות מתמטיות
- עבודה עם מחרוזות
- עיצוב התוצאות

למרות החשיבות שבתכנון ממשק המשתמש, כולל טפסים ופקדים, מרבית העבודה המעשית מתבצעת על ידי שימוש בקוד. במונח קוד הכוונה לקטעי תוכנה הכוללים משפטים ופונקציות. כבר ידוע לך שטפסים יכולים לכלול קוד, וכן הודגם איך ניתן לחדור אל הקוד "מתחת לפני השטח" של הממשק החזותי של הטופס. לשם תזכורת נסקור את כל חלקי תוכנת Visual Basic שייתכן שהם מכילים קוד:

- טפסים
- מודולי קוד
- מודולי מחלקה
- פקדי משתמש

יתרונה הגדול של Visual Basic הוא בכך שהשפה שבה נכתב הקוד היא גם בעלת עוצמה רבה וגם פשוטה יחסית לשימוש. היא צאצא ישיר לשפת התכנות הוותיקה BASIC. BASIC תוכננה כשפה למפתחים מתחילים, ומכאן שמה, המורכב מראשי התיבות "Beginner's All-Purpose Symbolic Instruction Code" ("קוד הדרכה סמלי לכל השימושים למפתח המתחיל").

פרק זה בוחן אחדים מהמושגים הבסיסיים בתכנות ב- Visual Basic: כתיבת קוד תוכנה, עבודה עם מידע עיצובי וכדומה. בפרק 10, **שליטה במהלך התוכנית**, נמשך הדיון בשאלת בקרת התוכנה בעזרת לולאות ומשפטים מותנים.

כתיבת משפטים

הפרק הקודם **שימוש במשתנים לאחסון מידע**, לימד מעט על משתנים וקבועים. הוסבר איזה סוג נתונים ניתן לאחסן בהם, וכיצד להצהיר עליהם. זוהי רק תחילת העבודה עם מידע בתוכנית. דרוש גם לדעת איך להקצות מידע למשתנים, להפעיל מידע זה ולהשתמש בתוכן המשתנים בקודים אחרים של Visual Basic.

לביצוע מטלות אלו משתמשים ב**משפטי קוד** (Statements). בהפשטה, משפט הוא שורת קוד, הגורמת למחשב לבצע פעולה. להלן דוגמאות אחדות לפעולות הניתנות לביצוע בעזרת משפטים:

סוג משפט	דוגמה
הקצה ערך למשתנה	sName = "Lauren"
קרא לפונקציה מוגדרת מראש	MsgBox "Good Morning!"
קרא לפונקציה שלך	bSuccess = BuildBudgetReport("12/31/99")
הגדר מאפייני אובייקט	frmMain.Visible = False
קבל החלטה	If nTideHeight > 1000 then MoveOn

הסעיפים הבאים דנים במשפטי הצבה בסיסיים ובפעולות ומחרוזות מתמטיות. בהמשך הפרק, בסעיף **תוצאות העיצוב** תלמד איך לעצב מספרים ומידע אחר המטופל בידי קוד התוכנית.

השימוש במשפטי הצבה

לאחר הכרזת משתנה, הפעולה הנדרשת ראשונה בדרך כלל, היא אחסון מידע באותו משתנה. זה תפקידו של **משפט ההצבה** (Assignment Statement). מבנה משפט ההצבה הוא פשוט: אתה מציין את המשתנה שאת ערכו ברצונך לקבוע, מציב לאחריו סימן שוויון, ולאחריו ביטוי המייצג את הערך האמור להיות מוצב בו. דוגמה למשפט הצבה בסיסי ביותר היא:

```
x = 5
```

משפט ההצבה אומר ל- Visual Basic להציב את המספר השלם 5 בשטח הזיכרון המיוצג על ידי המשתנה x. ההצבה אינה מוגבלת כמובן למספרים שלמים. הביטוי שאחרי סימן השוויון יכול להיות ערך מילולי, ביטוי או משוואה המורכבת משילוב כלשהו של משתנים וקבועים אחרים, או אפילו פונקציה מתמטית המגדירה ערך. אין הגבלה לדרגת המורכבות של הביטוי. המגבלה היחידה היא, שהביטוי חייב להחזיר ערך מסוג המשתנה אליו הוצב.

טבלה 9.1 מדגימה משפטי הצבה אחדים:

טבלה 9.1: סוגים של משפטי הצבה

סוג הביטוי	משפט הצבה
מחרוזת נומרית	nNumStudents = 25
מחרוזת מילולית	sTopStudent = "June Thomas"
ביטוי מתמטי	fAvgScore = nTotScore / nNumStudents
ביטוי מחרוזת	sSpouseName = "Mrs. " & "Tina Fortner"
ערך החזרה של פונקציה	sCapitalName = UCase$("Chris Cawein")

ייתכן שהבחנת בדמיון שבין ערכים אלה לערכים ששימשו לקביעת תכונות הטפסים והפקדים בסעיף **אופן ההפנייה לטפסים ופקדים בקוד**, שבפרק 3, **אבני היסוד של Visual Basic**. למעשה ערכים אלה זהים. רוב מאפייני הטפסים והפקדים זהות. ניתן לקבוע אותם בשלב התכנון, וגם לשנותם בעת ההרצה, תוך שימוש במשפטי הצבה.

מאפייני בקרה משמשים בחלק הימני של המשפט להצבת ערך במשתנה, ובחלק השמאלי לשינוי ערך המאפיין. לדוגמה: נניח שקיימות שלוש תיבות טקסט בטופס: txtLastname (שם המשפחה), txtFirstname (שם פרטי), txtEntirename (שם מלא). כשמשתמש מזין שם פרטי ושם משפחה, כדאי שהשם המלא יוזן אוטומטית.

דרך אחת לבצע זאת היא לאחסן כל מידע מתיבת הטקסט במשתנה, כמתואר להלן :

```
Dim sFName As String
Dim sLName As String
sFName = txtFirstName
sLName = txtLastName
```

ניתן עכשיו לאחסן את צירוף השם הפרטי ושם המשפחה במשתנה אחר, ולהשתמש בו לעדכון txtEntireName (השם המלא) באופן הבא :

```
Dim sName As String
sName=txtFirstName & * * & txtLastName
txtEntireName = sName
```

כמובן, מכיון שניתן להתייחס לתיבות טקסט כאל משתנים (לפי מאפיין ברירת המחדל של הפקדים), ניתן להגיע לאותה תוצאה ללא הצהרת משתנים, לדוגמה השורה הבאה :

```
txtEntireName = txtFirstName & * * & txtLastName
```

שורת קוד זו יעילה יותר, מכיון שאינה דורשת קביעת מימדי המשתנים. אולם, במקרה שקוד ניגש שוב ושוב למאפיין פקד (כמו למשל ב- for loop), אחסון מאפיין במשתנה זמני יעיל לפעמים יותר.

שימוש בפעולות מתמטיות

אחת מפעילויות המפתח של תוכניות מחשב רבות הוא עיבוד נתונים נומריים. פעולות מתמטיות קובעות הוצאת חשבונות ללקוחות וערכי ריבית על חסכונות ועל חיובי כרטיסי אשראי, מחשבות ממוצעי ציונים ומייצרות קטעי מידע רבים נוספים. Visual Basic תומכת במספר אופרטורים מתמטיים, שניתן להשתמש בהם במשפטי תוכנית. פעולות אלו וסמלי Visual Basic המייצגים אותם מרוכזים בטבלה 9.2. הפעולות מתוארות בפירוט בסעיפים הבאים.

טבלה 9.2: פעולות מתמטיות וסמלי Visual Basic התואמים להם

פעולה	אופרטור
חיבור	+
חיסור	-
כפל	*
חילוק	/
חילוק ללא שארית	\
שארית חילוק (Modulus)	mod
העלאה בחזקה	^

ב- Visual Basic משמשות הפעולות המתמטיות ליצירת משוואות. אלו כוללות אופרטורים, משתנים וביטויים אחדים, כמו למשל בדוגמה הבאה:

```
(115 + txtAmount.txt) / 69 * 1.0825
```

הערה:

הערך החוזר מ-txtAmount.txt הינו מחרוזת טקסט, אך Visual Basic יודעת להפוך ערך זה למספר, כדי לא ליצור חיבור בין ערך נומרי לערך טקסטואלי.

יש לשים לב כי למרות שהביטוי בדוגמה הוא בר-תוקף, שורת הקוד כשלעצמה איננה מבצעת דבר. תוצאת הפעולה צריכה להישלח ליעד כלשהו, למשל להיות מוצבת כמשתנה, או להיות מוצגת על צג המחשב.

חיבור וחיסור

שתי הפעולות המתמטיות הפשוטות ביותר הן חיבור וחיסור. לכל מי שהשתמש במחשבון לביצוע פעולות אלו יש מושג טוב איך הן מתבצעות כשורת קוד מחשב.

תוכנית מחשב מאפשרת גמישות רבה יותר ממחשבון. פעולות אלו אינן מוגבלות לעבודה בביטויים מספריים (למשל 1, 15, 37.63, 105.2-). התוכנה מאפשרת לחבר או לחסר שניים או יותר ביטויים מספריים, משתנים נומריים, או כל פונקציה המחזירה ערך מספרי. כמו-כן ניתן, כמו במחשבון, לבצע כל צירוף של פעולות חיבור וחיסור. נבחן עתה איך בדיוק מתבצעות פעולות אלו.

השימוש באופרטור החיבור

אופרטור החיבור ב- Visual Basic הוא סימן הפלוס (+). השימוש הכללי באופרטור הוא כמוצג בשורת התחביר הבאה:

```
result = number1 + number2 [+ number3]
```

result הוא משתנה (או מאפיין פקד) המכילה את סכום המספרים. סימן השוויון מציין הצבת ערך למשתנה. *number1*, *number2* ו-*number3* הם הביטויים המספריים, המשתנים המספריים או הפונקציות שיש לסכם יחד. ניתן לחבר מספרים ככל הנדרש, אך כל זוג מספרים חייב להיות מופרד בסימן פלוס.

השימוש באופרטור החיסור

אופרטור החיסור הוא הסימן מינוס (-). התחביר זהה בסיסית לזה של פעולת החיבור:

```
result = number1 - number2 [- number3]
```

בניגוד לפעולת החיבור, שבה אין משמעות לסדר הנתונים, בחיסור המספר לימין סימן המינוס מחוסר מהמספר לשמאלו. במקרה של פעולה רב מספרית, מחוסר המספר השני מהראשון, המספר השלישי מחוסר מהתוצאה וכן הלאה, תוך מעבר משמאל לימין בטור המספרים.

לדוגמה במשוואה הבאה :

```
result = 15 - 6 - 3
```

המחשב מחסר ראשית 6 מ-15, כשהתוצאה 9. אחר הוא מחסר 3 מ-9 ומקבל 6 כתוצאה סופית עבור המשתנה result.



טיפ:

ניתן לשלוט בסדר הפעולות על ידי שימוש בסוגריים. למשל: שורת הקוד הבאה תציב את הערך 12 במשתנה result:

```
result = 15 - (6 - 3)
```

ניתן ליצור משפטי הצבה הכוללים רק אופרטורים של חיבור או חיסור. ניתן גם להשתמש בשני האופרטורים יחד, או לשלבם עם אופרטורים מתמטיים נוספים. דוגמאות אחדות לאופרטורים תקפים הן :

```
val1 = 1.25 +.17
val2 = 3.21 - 1
val3 = val2 + val1
val4 = val3 + 3.75 - 2.1 + 12 - 3
val4 = val4 + 1
```

למי שאיננו מכיר תכנות מחשבים, ייתכן שהשורה האחרונה (שבה מופיע אותו שם משתנה גם לשמאל סימן השוויון וגם לימינו), נראית משונה במקצת. שורה כזאת אכן איננה מותרת בכמה שפות תכנות. ב- Visual Basic על כל פנים, ניתן לכלול שורת קוד האומרת לתוכנה לקחת את הערך הנוכחי של משתנה, להוסיף לו מספר אחר ולאחסן את התוצאה המתקבלת באותו משתנה.

כפל וחילוק

שני אופרטורים מתמטיים עיקריים נוספים שיש להכיר, הם כפל וחילוק. כמו חיבור וחיסור, נעשה בהם שימוש רב בחיי יום-יום.

השימוש באופרטור הכפל

ב- Visual Basic פעולת הכפל מבוצעת באופן ישיר, ממש כמו חיבור וחיסור. אתה פשוט משתמש באופרטור הכפל - הכוכבית (*) - להכפלת שני מספרים או יותר. תחביר משפט ההכפלה המופיע להלן, זהה כמעט לזה של משפט החיבור והחיסור :

```
result1 = number1 * number2 [* number3]
```

כמו במקרים הקודמים, result1 הוא שם המשתנה שנבחר לאחסון תוצאות ההכפלה, ואילו *number1*, *number2* ו-*number3* הם ביטויים מספריים, משתנים נומריים או פונקציות.

השימוש באופרטורים של חילוק

פעולת החילוק ב- Visual Basic מורכבת מעט יותר מכפל. הקוד שבתוכנית 9.1 להלן מציג את השימוש בשיטת חילוק אחת, זו המוכרת בחיי היומיום ומשמשת במחשבונים. התוצאה מתקבלת כמספר והשבר העשרוני שלו (אם יש שבר).

אולם, זו היא רק אחת משלוש שיטות חילוק הנתמכות על ידי Visual Basic. הן מוכרות כ**חילוק עם נקודה צפה** (Floating point division, השיטה הרגילה והמוכרת), **חילוק מספר שלם** (Integer division) ו**חילוק מודולוס או שארית חילוק** (Modulus, or remainder, division).

חילוק עם נקודה צפה הוא החילוק הטיפוסי, שמלמדים בבית הספר. מספר אחד מחולק בשני, והתוצאה היא מספר עשרוני. האופרטור שלו הוא הקו הנטוי (/):

```
result = number1 / number2 [/ number3]
```

השורה הבאה מחזירה את התוצאה 1.33333:

```
Print 4 / 3
```

חילוק מספר שלם מחלק מספר אחד בשני, ומחזיר רק את החלק השלם של התוצאה. האופרטור שלו הוא הקו הנטוי ההפוך (\):

```
result = number1 \ number2 [\ number3]
```

השורה הבאה מחזירה את התוצאה 1:

```
Print 4 \ 3
```

חילוק מודולוס (או שארית חילוק) מחלק מספר אחד בשני, ומחזיר כתוצאה את השארית המתקבלת לאחר קבלת מנת השלם הגדולה ביותר האפשרית. אופרטור הפעולה הוא המילה הלטינית mod:

```
result = number1 mod number2 [mod number3]
```

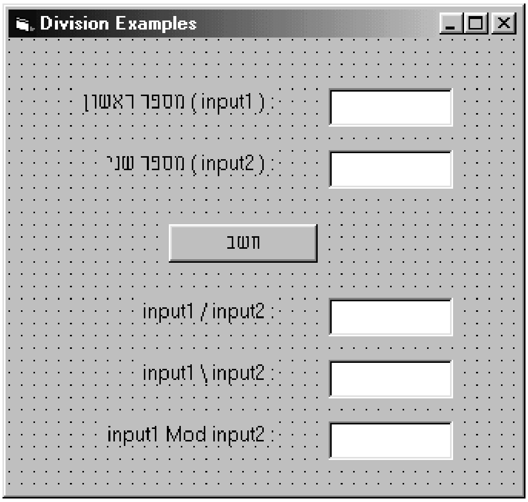
השורה הבאה מחזירה את התוצאה 2, השארית המתקבלת כשמחלקים 20 ב-3:

```
Print 20 mod 3
```

כמו בחיבור, חיסור וכפל, אם מתבצע חילוק של יותר משני מספרים, כל זוג מספרים צריך להיות מופרד באופרטור חילוק. כמו כן, כמו במקרים האחרים, פעולה מרובת אופרטורים מתבצעת על ידי קריאת המשוואה משמאל לימין.

תרשים 9.1 מדגים את ההבדלים בין האופרטורים השונים של החילוק. הקוד ללחצן הפקודה בטופס המוצג הוא:

```
input1 = Text1.Text
input2 = Text2.Text
Text3.Text = input1 / input2
Text4.Text = input1 \ input2
Text5.Text = input1 mod input2
```

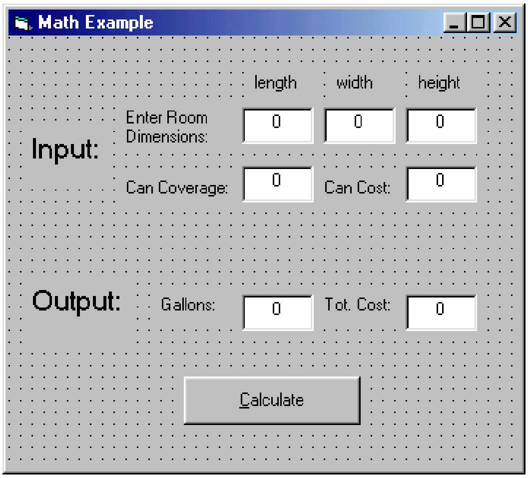


תרשים 9.1: תוכנית זו מדגימה את ההבדלים בין שלושה סוגי פעולות החילוק של Visual Basic

לאחר בניית הטופס הרץ את התוכנית. הצב 5 בתיבת הטקסט הראשונה ו-3 בשנייה, ולחץ על לחצן הפקודה. שים לב למספרים השונים המופיעים בתיבות הטקסט המציגות תוצאות. ניתן לנסות דוגמה זו גם בצירופי מספרים אחרים.

השימוש בכפל ובחילוק בתוכנית

כהדגמה לשימוש בכפל ובחילוק בתוכנית מחשב מובאת להלן דוגמת תוכנה לקביעת כמות הצבע הדרושה לצביעת חדר. תוכנה כזאת יכולה לכלול טופס, המאפשר לצבעי להזין את אורך החדר ורוחבו, גובה התקרה, שטח הכיסוי של פחית צבע אחת ואת מחירה. דוגמת הטופס מופיעה בתרשים 9.2, והקוד לביצוע החישובים מופיע ב-MATHEX.VBP.



תרשים 9.2: כדי לקבוע את כמות הצבע הדרושה לצביעת חדר משתמשים בפעולות כפל וחילוק

תוכנית 9.1: MATHEX.VBP הערכת עלות צביעה בשימוש באופרטורים של כפל וחילוק

```
Private Sub cmdCalculate_Click()
    Dim fRoomLength As Single
    Dim fRoomWidth As Single
    Dim fRoomHeight As Single
    Dim cCanCoverage As Single
    Dim cCanCost As Currency
    Dim fRoomPerimeter As Single
    Dim fWallArea As Single
    Dim fNumGallons As Single
    Dim cProjCost As Currency

    fRoomLength = Val(txtLength.Text)
    fRoomWidth = Val(txtWidth.Text)
    fRoomHeight = Val(txtHeight.Text)
    fCanCoverage = Val(txtCoverage.Text)
    cCanCost = Val(txtCost.Text)
    fRoomPerimeter = 2 * fRoomLength + 2 * fRoomWidth
    fWallArea = fRoomPerimeter * fRoomHeight
    fNumGallons = fWallArea / fCanCoverage
    cProjCost = fNumGallons * cCanCost
    txtGallons.Text = fNumGallons
    txtTotalCost.Text = cProjCost

End Sub
```

כדי לנסות קוד זה, יש להתחיל ביצירת פרויקט EXE ב- Visual Basic. הוסף לטופס שבע תיבות טקסט. קבע את מאפייני Name שלהן ל- txtLength, txtHeight, txtCoverage, txtGallons ו-txtTotalCost. חמש תיבות הטקסט הראשונות מיועדות לקלוט מידע מהמשתמש. שתי האחרונות מחזירות מידע. ייתכן שתרצה להוסיף פקדי רמה תיאורית לצד כל תיבת טקסט.

בשלוש תיבות הטקסט העליונות שבדוגמה מזינים את מימדי החדר. בשתי תיבות הטקסט האמצעיות מזינים את יכולת הצבע לכסות במ״ר ואת עלותו. שתי תיבות הטקסט התחתונות מציגות את מספר הגלונים שנדרש לביצוע הצביעה ואת העלות.

אחר יש להוסיף לטופס לחצן פקודה. שנה את מאפיין Name שלו ל-cmdCalculate, ואת מאפיין Caption ל- Calculate Job Cost. הוסף את שורות הקוד המופיעות בתוכנית 9.1. שמור והרץ, כדי לראות את הפעולה.

שימוש במעריך

מעריכים נקראים גם **חזקות**. למשל, 2 בחזקת שלוש (2^3), שווה-ערך ל- 2X2X2, או 8. מעריכים משמשים רבות בפעולות מחשב, שבהן ערכים רבים מיוצגים כביטוי בחזקת שתיים. הם גם מופיעים בהרחבה בעבודות מדעיות והנדסיות, שבהן מונחים מתמטיים מוצגים לעיתים קרובות כחזקות של 10 או כלוגריתמים טבעיים. מעריכים פשוטים יותר משמשים בסטטיסטיקה, שבה חישובים רבים תלויים בריבועי מספרים או בשורשיהם הריבועיים.

כדי להעלות מספר בחזקה משתמשים ב**אופרטור המעריכי** (Exponential Operator), או בחץ העילי (^). מעריכים גדולים מיחידה מייצגים העלאה בחזקה. מעריכים שהם שבר מייצגים הוצאת שורש, ומעריכים שליליים מייצגים שברים. להלן תחביר האופרטור המעריכי:

```
answer = number1 ^ number2
```

המשוואות בטבלה הבאה מציגות שימושים נפוצים במעריכים. מצוינות גם הפעולות שכל משוואה מבצעת.

פעולה מבוצעת	דוגמת מעריך
העלאת מספר בריבוע	3 ^ 2 = 9
הוצאת שורש ריבועי ממספר	9 ^ 0.5 = 3
הצגת שבר ע"י מעריך שלילי	2 ^ -2 = 0.25

קדימות אופרטורים

ביטויים רבים מכילים שילוב של האופרטורים שתוארו. במקרים כאלה חשוב לדעת באיזה סדר מפעילה Visual Basic את טיפוסי האופרטורים השונים. למשל, מה ערך הביטוי 2 / 6 + 3 * 4? ניתן לחשוב שהאופרטורים מופעלים לפי סדר הופעתם משמאל לימין. במקרה זה 3 * 4 שווים 12, 6 + 12 שווים 18, 2 / 18 שווים 9. למעשה, Visual Basic אינה מפעילה בהכרח את האופרטורים לפי סדר הופעתם משמאל לימין. היא עובדת לפי סדר הפעלה ברור, הידוע כ**קדימות אופרטורים** (Operator Precedence).

Visual Basic מפעילה תת-קבוצות של אופרטורים לפי סדר הקדימויות הבא:

- פעולות מעריכיות (^)
- הפעלת סימן שלילי (-)
- כפל וחילוק (*, /)
- חילוק מספר שלם (\)
- חישובי מודולוס (mod)
- חיבור וחיסור (+, -)

בתוך תת-קבוצה מופעלים האופרטורים לפי סדר הופעתם משמאל לימין. לאחר הפעלת כל תת-קבוצה, מחושבת שארית הביטוי משמאל לימין.

בדוגמה הקודמת (2 / 6 + 3 * 4), קטעי הכפל והחילוק (3 * 4 השווים 12 ו- 2 / 6 השווים 3) יחושבו תחילה, ואחר יתבצע החישוב הפשוט יותר של 3 + 12, ששוויו 15.



אזהרה

הבנת סדר קדימויות האופרטורים חיונית כדי להבטיח שתוכנית תבצע את החישובים כמצופה ממנה. למשל: נניח שיש לחשב את ממוצע התוצאות של שתי קבוצות ניסויים. ניתן לכתוב את שורת הקוד הבאה:

```
fAvgScore = nTest1 + nTest2 / 2
```

שורת קוד זו נראית לכאורה נכונה, אך חישובי Visual Basic לא יהיו מדויקים. מכיון שלאופרטור החילוק קדימות גבוהה מאופרטור החיבור, תת-ביטוי nTest2 / 2 יחושב ראשון, והערך המחושב יתווסף ל-nTest1. ברור שזה איננו סדר הפעולות הנכון. בשימוש בסוגריים ניתן למנוע את הבעיה ולשלוט על סדר ביצוע הפעולות:

```
fAvgScore = (nTest1 + nTest2) / 2
```

ביטוי זה יחושב כנדרש. הביטוי (nTest1 + nTest2) יחושב ראשון, והסכום יחולק ב-2.

נניח ש-nTest1 שווה 97 ו-nTest2 הוא 88. הביטוי מחושב לפי 2 / (88 + 97), או 2 / 185, המחזיר ל-fAvgScore את הערך הנכון של 92.5. השמטת הסוגריים תביא (לפי סדר קדימויות האופרטורים) את התוצאה של 2 / 88 + 97, או 44 + 97, או 141.

הערה חשובה:

אפשר לשנות את סדר קדימויות האופרטורים, על ידי שימוש בסוגריים ליצירת תת-ביטויים שיחושבו קודם. ניתן גם לבנות מערכות של סוגריים בתוך סוגריים. Visual Basic מחשבת ראשית ביטויים בתוך הסוגריים, פנימיים לפני חיצוניים, ולאחר מכן מפעילה את סדר קדימויות האופרטורים.

הנושא **Operator Precedence** שבעזרה של Visual Basic מציג דיון טוב בנושא, לרבות השאלה על הפעלת הקדימויות באופרטורים של השוואה ואופרטורים לוגיים.

עבודה עם מחרוזות

בעבודתו משתמש המפתח במחרוזות למטרות רבות. ככל שתפעול המחרוזות נעשה בצורה יעילה יותר, התוכנית נראית מקצועית יותר. Visual Basic מאפשרת גמישות רבה בכל הקשור לשימוש במחרוזות. היא תומכת באופרטור מחרוזות אחד בלבד - אופרטור **השרשור** (Concatenation), אך פונקציות מחרוזת אחדות מובנות לתוך Visual basic, וניתן להשתמש בהן:

- UCase ו-LCase - שנה את כתיב הטקסט לאותיות רישיות (Uppercase) או לקטנות (Lowercase) בהתאמה.
- InStr ו-InStrRev - מצא מיקום מחרוזת אחת הממוקמת בתוך מחרוזת אחרת.
- Left ו-Right - שלוף מספר תווים נתון מצידה הימני או השמאלי של המחרוזת.
- Mid - שלוף או החלף מספר תווים נתון במחרוזת.
- LTrim, RTrim ו-Trim - סלק מרווחים מקצה אחד או משני קצות המחרוזת.
- Len - מחזירה אורך מחרוזת (לפי מספר תווים).

הערה:

כמה מהפונקציות שברשימה (למשל UCase) מחזירות ערך מטיפוס Variant. לגבי כל אחת מהן, פונקציה בשם זהה עם סימן דולר ($) בסוף השם מצביעה שהערך המוחזר הוא הסוג מחרוזת. מומלץ להשתמש בגרסת $ בכל מקום שזה אפשרי (למשל Left$), כי היא יותר יעילה.

בקטע זה נבחן כיצד ניתן לתפעל מחרוזות, כדי לספק לתוכנית הופעה מקצועית.

שרשור מחרוזות

כמצוין לעיל, קיים ב- Visual Basic אופרטור מחרוזות אחד - **אופרטור השרשור** (Concatenation Operator). האופרטור, המסומן בתו &, מחבר שתי מחרוזות טקסט או יותר, בדומה לאופן שאופרטור החיבור מחבר שני מספרים או יותר. בצירוף שתי מחרוזות בעזרת אופרטור השרשור, המחרוזת השנייה מחוברת ישירות לקצה המחרוזת הראשונה. כתוצאה מתקבלת מחרוזת הכוללת את התוכן המלא של שתי מחרוזות המקור. אופרטור השרשור משמש כמשפט הצבה באופן הבא:

```
newstring = stringexpr1 & stringexpr2 [& stringexpr3]
```

בתחביר זה newstring מייצג את המשתנה המכיל את תוצאת פעולת השרשור. stringexpr1, stringexpr2 ו-stringexpr3 מייצגים כולם ביטויי מחרוזת. אלה יכולים להיות כל מחרוזת אפשרית, כולל משתני מחרוזת, מחרוזות מילוליות (מוקפות במרכאות), או פונקציות המחזירות מחרוזת. התו & בין צמד ביטויי מחרוזת מורה ל- Visual Basic לרכז את שני הביטויים. לפני התו & ואחריו חייב להופיע רווח. התחביר מציג תו & שני אופציונלי וכן מחרוזת שלישית. ניתן לחבר כל מספר של מחרוזות, כל זמן שכל זוג מופרד על ידי התו &.

הערה:

בהסבת תוכניות מגרסאות ישנות של Visual Basic,יש לפעמים מחרוזות המחוברות בעזרת (+). שימוש כזה היה שכיח בגרסאות מוקדמות של Visual Basic, כמו בשפות BASIC ישנות. למרות ש- Visual Basic עדיין תומכת בשימוש בסימן פלוס לשרשור מחרוזות (במקרה שאופרטור זה מופיע בגרסאות ישנות העוברות התאמה), ההמלצה היא להשתמש בתו & למטרה זו. כך נמנע הבלבול בין פעולת השרשור לחיבור.

תוכנית 9.2 מראה איך לשרשר מחרוזות בתוכנית פשוטה, ליצירת תווית כתובת. השדות מתיבות טקסט שונות מחוברים כדי ליצור שורות שונות לצורך התווית.

תוכנית 9.2: MAILING.VBP - השימוש בשרשור מחרוזות לשם יצירת תוויות כתובת

```
Private Sub cmdPrintAddr_Click()
    Dim sFName As String, sLName As String
    Dim sAddr As String, sCity As String
    Dim sState As String, sZIP As String
    Dim sTitle As String
    Dim sName As String, sCSZ As String
    sFName = txtFirst.Text
    sLName = txtLast.Text
    sAddr = txtAddress.Text
    sCity = txtCity.Text
    sState = txtState.Text
    sZIP = txtZIP.Text
```

```
        If optMr.Value Then sTitle = “Mr. ”
        If optMrs.Value Then sTitle = "Mrs. "
        If optMiss.Value Then sTitle = "Miss "
        If optMs.Value Then sTitle = "Ms. "
        sName = sTitle & sFName & " " & sLName
        sCSZ = sCity & ", " & sState & " " & sZIP
        Printer.Print sName
        Printer.Print sAddr
        Printer.Print sCSZ

End sub
```

הטופס לתוכנית זו מוצג בתרשים 9.3.



תרשים 9.3: היישום תווית כתובת מציג כיצד אפשר לחבר מחרוזות לתצוגה או הדפסה

קביעת אורך המחרוזת

בפעולות רבות יש צורך לדעת כמה תווים נכללים במחרוזת. למשל: כדי לדעת אם מחרוזת תתאים לשדה בעל אורך נתון בבסיס נתונים, או, במקרה של עבודה עם מחרוזות ארוכות, יש לפעמים צורך לוודא שאורכן המשולב של שתי מחרוזות מחוברות אינו חורג מקיבולת משתנה המחרוזת. בכל מקרה, לקביעת אורך מחרוזת כלשהי, ניתן להשתמש בפונקציה Len, כמודגם בשורת הקוד הבאה:

```
result = Len (inputstr)
```

לפונקציה Len יש כמה שימושים. במקרים רבים משתמשים בה כדי לקבוע אם במחרוזת מסוימת מופיעים תווים. במקרה שהמחרוזת ריקה, ייתכן שיהיה צורך לייצר הודעת שגיאה או לעקוף עיבודים נוספים.

שינוי גודל האותיות במחרוזת

שתי פונקציות משמשות לשינוי גודל אותיות במחרוזת: UCase ו-LCase. הפונקציה UCase מחזירה מחרוזת שכל אותיותיה שונו לרישיות (Uppercase), פונקציית LCase עושה את ההפך: מחזירה מחרוזת שכל אותיותיה קטנות (Lowercase).

פונקציות אלו נראות טריוויאליות, אך הן שימושיות למדי לביצוע כמה מטלות. ראשית ניתן להשתמש בהן להבטחת הצבה נכונה של אותיות רישיות. הקוד בתוכנית 9.3 מציב אות רישית בראש המילה, ואותיות קטנות בהמשך:

תוכנית 9.3: CASECONV.VBP שימוש ב-UCase וב-LCase להצבה נכונה של אותיות רישיות במילים לטיניות.

```
Dim sWord as String, sProperWord as String
sWord="mIxEd CaSe"
sProperWord = UCase$(Left$(sWord, 1))
sProperWord = sProperWord & LCase$(Mid$(sWord,2))
```

הערה:

פונקציות מחרוזת מחזירות ערכים, אך אינן משנות את הקלט. למשל הדוגמה:

```
Dim s1 As String
Dim s2 As String
s1 = "which case am I"
s2 = UCase$(s1)
```

לאחר הרצת הקוד, המשתנה s2 מופיע כולו באות רישית, בעוד s1 נשאר ללא שינוי, אלא אם כן s1 מוצב בשני צידי משפט ההצבה:

```
s1 = UCase$(s1)
```

פונקציות Case שימושיות למשל לצורך השוואה בין נתוני קלט של משתמש, לטווח ערכים מוגדר מראש. אם נתוני קלט מומרים לאותיות רישיות, ניתן להשוותם למחרוזת בדיקה המורכבת מאותיות גדולות:

```
Select Case UCase(txtOperation.Text)

        Case "WASH"
                ' Do Something
        Case "RINSE"
                ' Do Something else
        Case "SPIN"
                ' Do Something else Yet
         Case Else
                MsgBox "Invalid Entry!"
End Select
```

בקוד זה, אם לא נעשה שימוש בפונקציה UCase, התשובה Invalid Entry היתה מוחזרת גם אם נעשתה הבחירה "הנכונה" של אותיות קטנות או מעורבות (למשל המחרוזת Rinse).

טיפ:

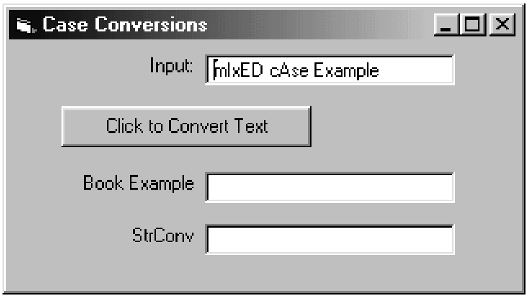
בתכנות מקובל להשתמש בקינון פונקציות מחרוזת בתוך שורות קוד, במקום השימוש במשתנים נוספים לצורך אחסון נתוני כל צעד ביניים. נבחן לדוגמה את השורה Select Case בקוד המובא להלן, העושה שימוש בכמה פונקציות מחרוזת מקוננות לעזרה באימות קלט. המשתמש יכול להשתמש כרצונו בכל צירוף אותיות קטנות וגדולות, והקוד יטפל בה:

```
stUserInput = InputBox$("Type Yes or No, please!")
Select Case Left$(Trim$(Ucase$(stUserInput)),1)
        Case "Y"
                MsgBox "Yes"
        Case "N"
                MsgBox "type YES or No please!"
End Select
```

פונקציה אחרת של Visual Basic, StrConv, מבצעת המרות מיוחדות של מחרוזות. רוב ההמרות שהיא מסוגלת לבצע הן עודפות וחוזרות על עצמן (למשל הפיכת כל המחרוזת לאות רישית או לאות קטנה), או חורגות ממסגרת ספר זה (למשל המרה בין סוגי אותיות יפניות), אבל כדאי להזכיר כאן סוג המרות אחד כזה. הפונקציה StrConv מסוגלת להסב משפט למצב **proper case**, שבו האות הראשונה בכל מילה היא אות רישית. דוגמת הקוד הבאה מדגימה את הטכניקה:

```
lblHeadline = StrConv(stHeadline, vbProperCase)
```

דוגמאות לשימוש בפונקציות UCase, LCase וכן StrConv מופיעות בתרשים 9.4.



תרשים 9.4: תוכל להשתמש ב-LCase, UCase ו-StrConv כדי לשנות את גודל האותיות במחרוזת טקסט

חיפוש מחרוזת

במשימות רבות הקשורות במחרוזות, דרישת התכנות הראשונה היא לקבוע אם והיכן מופיעה מילה מסוימת, ביטוי או צירוף תווים אחר בתוך מחרוזת. היכולת למצוא מחרוזת אחת בתוך אחרת מאפשרת חיפוש מילה בתוך טקסט. בחיפוש כזה ניתן לבצע החלפה גלובלית של מחרוזת, לדוגמה החלפת המילה "טקסט" במילה "מחרוזת" בכל מקום שהיא מופיעה במסמך נתון.

סיבה אחרת, נפוצה יותר, לחיפוש בתוך מחרוזת, הוא הצורך ב**ניתוח מבנה משפט** (Parsing). ניקח כדוגמה את משפט הקלט הבא, הכולל שם של אדם במבנה כדלהלן: "Dr. Stirling P. Williams, Jr.". בתיק הכולל מאות מחרוזות כאלו, אחסון המידע בבסיס נתונים, עם הפרדה בין שדה השם הפרטי ושם המשפחה, נראה קצת מסובך. אבל ניתן להשתמש בפונקציות חיפוש מחרוזות ובלוגיקת תכנות, כדי לנתח את מבנה המשפט ולפרקו לחלקים קטנים יותר.

הפונקציה המאפשרת חיפוש תו או קבוצת תווים בתוך מחרוזת נקראת InStr. יש לה שני פרמטרי חובה ושני פרמטרי רשות. שני הפרמטרים הנדרשים הם המחרוזת הנבדקת והטקסט אותו יש לחפש. אם הטקסט המבוקש מופיע במחרוזת, InStr מחזירה את אינדקס התו שבו מתחיל הטקסט המבוקש. אם הטקסט איננו נמצא במחרוזת, מחזיר האינדקס את התו 0. התחביר הפשוט של פונקציית InStr מוצג להלן:

```
chrpos = InStr(sourcestr, searchstr)
```

לדוגמה הפונקציה:

```
Print Instr("I'll see you next Tuesday.","you")
```

מחזירה את התוצאה 10, מכיון שזה מיקום תחילת המילה **you**.

פרמטר הרשות הראשון של הפונקציה InStr מציין לה את מיקום התו הראשון שבו יש להתחיל בחיפוש. פרמטר זה חייב להיות מספר חיובי שלם. אם הפרמטר גדול מאורך המחרוזת, InStr תחזיר את הערך 0. תחביר הפונקציה הוא:

```
chrpos = InStr(StartPos, sourcestr, searchstr)
```

לדוגמה הפונקציה:

```
Print Instr(7,"Pride cometh before a fall","e")
```

מחזירה את הערך 10, למרות שה-e הראשון מופיע במקום 5, מכיון שהחיפוש מתחיל במיקום השביעי.

פרמטר הרשות השני קובע אם החיפוש יהיה תלוי-רישיות. מתן ערך 0 לפרמטר ההשוואה (ערך המחדל שלו), מבצע חיפוש תלוי-רישיות. הצבת ערך 1 לפרמטר מבצע חיפוש שאינו תלוי-רישיות. התחביר הוא:

```
chrpos = InStr(StartPos, sourcestr, searchstr, 1)
```

בעזרת פרמטרי הרשות אפשר לכתוב קוד המוצא בזה אחר זה את כל מחרוזות החיפוש בטקסט הנבדק. הקוד בתוכנית 9.4 מדפיס את המילים במחרוזת, המופרדות ברווחים. הוא פועל על ידי לקיחת תוצאת הפונקציה Instr והעברתה בחזרה לפרמטר StartPos.

תוכנית 9.4: INSTREX.VBP - שימוש בפונקציה לחלוקת מחרוזת למילים

```
Sub PrintWords(stInput As String)
    Dim inCounter As integer
    Dim inFoundPos As Integer

    Const PARSECHAR = " " 'Space

    'If string is blank then do nothing
    If Len(stInput) = 0 Then Exit Sub

    'Search for a space
    inFoundPos = InStr(incounter, stInput,PARSECHAR)

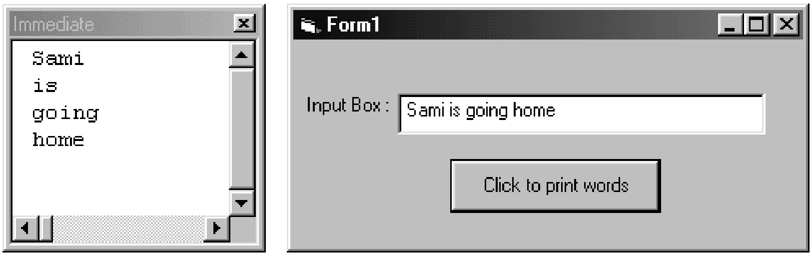
    'If a space is found print the word and keep searching
    While inFoundPos <> 0
       Debug.Print Mid$(stInput, inCounter, inFoundPos - inCounter)
       inCounter = inFoundPos + 1
       inFoundPos = InStr(inCounter, stInput, PARSECHAR)
    Wend

    'Print the remainder of the string
    If inCounter < Len(stInput) Then
       Debug/Print Mid$(stInput, inCounter)
    End If
End Sub
```

הקלט ותוצאות הקוד הזה מופיעים בתרשים 9.5.

בתוכנית 9.4 מבצעת הפונקציה InStr חיפוש מהתו הראשון במחרוזת עד התו האחרון. ב- Visual Basic קיימת פונקציה דומה, InstrRev, המבצעת את החיפוש בכיוון ההפוך. הפונקציה עובדת כמו InStr, אך התחביר שונה במקצת:

```
InstrRev(string1, string2[, start[, compare]])
```



תרשים 9.5: השתמש בפונקציה InStr כדי למצוא את כל הרווחים במחרוזת

ההבדל העיקרי הוא שפרמטר ההתחלה נמצא **אחרי** המחרוזת, לא לפניה. בשימוש ב-InstrRev ניתן לשנות קלות את הקוד שבתוכנית 9.4 ולבנות תת-שיגרה ששמה PrintWordsReverse:

```
Sub PrintWordsReverse(stInput As String)
    Dim inCounter As Integer
    Dim inFoundPos As Integer

    Const PARSECHAR = " " 'Space

    ' If string is blank then do nothing
    If Len(stInput) = 0 Then Exit Sub

    ' Start at the last character
    inCounter = Len(stInput)

    ' Search for a space
    inFoundPos = InStrRev(stInput, PARSECHAR, inCounter)

    ' Print the remainder of the string
    If inCounter > 0 Then
       Debug.Print Left$(stInput, inCounter)
    End if
End Sub
```

חילוץ חלקי מחרוזת

נבחן שוב את הקוד שבתוכנית 9.3. בנוסף ל-UCase ו-LCase נעשה שימוש בפונקציות אחדות לחילוץ קטעי טקסט מהמחרוזת המקורית.

במצבים רבים יש צורך לעבוד רק עם חלק של מחרוזת. לדוגמה: יש צורך לחלץ את שמו פרטי של אדם מתוך השם המלא, או שדרוש לוודא שהמידע המטופל מתאים לשדה בבסיס הנתונים בו הוא אמור להיות מאוחסן. ניתן לבצע זאת בקלות בעזרת הפונקציות הבאות של Visual Basic:

- Right - מחלץ מספר תווים נתון מהקצה הימני של מחרוזת.
- Left - מחלץ מספר תווים נתון מהקצה השמאלי של מחרוזת.
- Mid - מחלץ תווים ממרכז מחרוזת.

נבחן ראשית את הפונקציות Left ו-Right, הקלות יותר לשימוש. (אגב, השימוש בשום פונקציה מהנדונות כאן איננו קשה במיוחד). כדי להשתמש בהן, יש לקבוע את מחרוזת הקלט וכן את מספר התווים שיישלפו. תחביר שתי הפונקציות הוא:

```
OutStr = Left$(InptStr, NumChars)
OutStr = Right$(InptStr, NumChars)
```

בשימוש בפונקציות אלו מספר התווים המצוין חייב להיות גדול מאפס או שווה לו. אם מצוין המספר 0, המחרוזת המוחזרת היא באורך אפס. אם המספר גדול מאורך המחרוזת, מוחזרת המחרוזת בשלמותה.

בכתיבת תוכניות המטפלות במחרוזות, הפונקציות Left ו-Right מופיעות לעיתים קרובות בצירוף עם פונקציות מחרוזת אחרות. תוכנית 9.5, המחלצת את שמו הפרטי של אדם משמו המלא, מדגימה את הנושא. הפונקציה מניחה שהשם הפרטי ושם המשפחה מופרדים ברווח. הפונקציה מחפשת את הרווח בטקסט הפלט, מחלצת את התווים הקודמים לרווח ומחזירה אותם כשם פרטי. אם אין רווחים במחרוזת, הפונקציה מניחה שזו כוללת רק את השם הפרטי.

תוכנית 9.5: INTREX.VBP - שימוש בפונקציות InStr ו-Left לחילוץ שם פרטי של אדם

```
Dim sName As String
Dim sFirstName As String
Dim nSpacePos as Integer

sName = Trim$(txtName.Text)
nSpacePos = InStr(sName, " ")

If nSpacePos > 0 then
    sFirstName = Left$(sName, nSpacePos - 1)

    lblname = "hello, my name is " & sFirstName
End If
```

Mid היא פונקציה נוספת המשמשת לחילוץ תת-מחרוזות ממחרוזות. היא פועלת בדרך דומה ל-Left ו-Right, אך כוללת ארגומנט נוסף. הפונקציה משמשת לחילוץ תו, מילה או ביטוי מתוך מחרוזת.

ל-Mid שלושה ארגומנטי חובה וארגומנט רשות אחד כמוצג בשורת התחביר הבאה:

```
newstr = Mid(sourcestr, startpos[, numchars])
```

startpos מייצג את מיקום התו שממנו מתחילה החזרת תת-מחרוזת. אם startpos גדול מאורך המחרוזת, מוחזרת מחרוזת ריקה. ארגומנט הרשות numchars מציין את מספר התווים המוחזרים מ-sourcestr. אם numchars מושמט, תחזיר הפונקציה את תווי המחרוזת החל מנקודת ההתחלה עד לסופה, כמודגם להלן:

```
Print Mid("Robert Allen",8)
```

מחזירה את המחרוזת "Allen"

```
Print Mid("Robert Allen",8,2)
```

מחזירה את המחרוזת "Al"

סילוק רווחים

בתוכנית 9.5 משמשת InStr למציאת הרווח הראשון ברשומת שמו של אדם, בהנחה שזה הרווח בין שמו הפרטי לשמותיו האחרים. רוב המחרוזות הארוכות כוללות בתוכן רווחים, הנחוצים לצורך ריווח נכון של מילים, סעיפים וכדומה. אבל ישנם מקרים, שמחרוזות כוללות רווחים בלתי רצויים בתחילתן ובסופן. רווחים אלה באים בדרך כלל כתוצאה מהקלדה מקרית של מרווחים בתחילת שדות טקסט או בסופן. הם מופיעים גם בשימוש במחרוזות בעלות אורך קבוע מראש, שמספר התווים בהן איננו מספיק למלאן.

לדוגמה כל אחת מהפניות הבאות לפונקציה ()Len תחזיר מספר שונה:

```
Print Len("Hello, world!")
```

```
Print Len("  Hello, world!")
```

```
Print Len("Hello world!   ")
```

בדרך כלל רווחים אינם גורמים נזק, למעט תפיסת תאי זיכרון בודדים. לעומת זאת, במקרים של חיבור מחרוזות, וכן בפעולות המבוססות על תוכנן, רווחים בלתי רצויים עלולים להזיק. למשל: נניח שקיימות שתי תיבות טקסט, לשם הפרטי ולשם המשפחה, כל אחת באורך שלושים תווים. משתמש עלול להקליד בהן בטעות שלושה תווים, ואחריהם קבוצה שלמה של רווחים מיותרים. אם דרוש לרכז את השם הפרטי ושם המשפחה, למשל לשם הכנת מדבקת כתובת, הרווחים הנוספים ייכללו במדבקה. ב- Visual Basic קיימים מספר כלים המאפשרים "קיצוץ" רווחים עוקבים מיותרים.

לסילוק רווחים בסוף מחרוזת ניתן להשתמש בפונקציות Visual Basic הבאות:

- LTrim - מסלק רווחים מצידה השמאלי של המחרוזת.
- RTrim - מסלק רווחים מצידה הימני של המחרוזת.
- Trim - מסלק רווחים גם מתחילת מחרוזת וגם מסופה.

כל אחת מפונקציות אלו משתמשת בתחביר דומה. דוגמה כבר הוצגה בתוכנית 9.5.

הקוד הבא מדגים את השימוש בפונקציה Trim לסילוק הרווחים בדוגמה של תווית הכתובת:

```
picMail.Print Trim(FirstName)
picMail.Print Trim(Address)
picMail.print Trim(City) & ", " & " " & Trim(Zip)
```

החלפת תווים במחרוזת

נוסיף עכשיו מעט בלבול לסיפור. בסעיף הקודם הודגם איך הפונקציה Trim שולפת קטע מחרוזת מאמצע מחרוזת המקור. אותה מילת מפתח Trim משמשת להשבת קטע ממחרוזת. אולם במקרה זה התחביר שונה, בכך שהפונקציה פועלת על חלקו השמאלי של משפט ההצבה. המשפט המשמש להחלפת תווים במחרוזת נקרא משפט Mid.

משפט Mid, המחליף חלק ממחרוזת אחת במחרוזת אחרת, משתמש בתחביר הבא:

```
Mid(sourcestr, startpos[, numchars]) = replstr
```

במקרה זה sourcestr היא המחרוזת המקבלת את תווי ההחלפה. sourcestr חייבת להיות משתנה מחרוזת, לא מחרוזת מילולית ולא פונקציית מחרוזת. startpos הוא מיקום התו שממנו תתחיל ההחלפה. מיקום זה חייב להיות מספר שלם גדול מאפס. numchars הוא ארגומנט רשות, המציין את מספר התווים ממחרוזת ההחלפה שבהם תשתמש הפונקציה. replstr מייצג את המחרוזת המכילה את התווים המוחלפים. הוא יכול להיות משתנה מחרוזת, מחרוזת מילולית, או פונקציית מחרוזת.

משפט Mid משמר את אורכה המקורי של המחרוזת. במילים אחרות, אם המרווח שבין נקודת תחילת ההחלפה לבין קצה המחרוזת קצר מאורך המחרוזת המחליפה, יועתקו רק תווים כאורך המרווח, החל משמאל המחרוזת המחליפה.

יש למשפטי Mid כמה שימושים. בתוכנית 9.3 לעיל הוצגה דוגמה להצבת אותיות רישיות. בשימוש במשפטי Mid, ניתן לבצע את אותה ההצבה בעזרת הקוד הבא:

```
Dim sWord as String, sProperWord as String
sWord="mIxEd CaSe"
sProperWord = LCase$(sWord)
Mid(sProperWord,1) = UCase$(Left$(sWord,1))
```

בתוכנית אחרת היה צורך לסלק את כל סימני הורדת שורה (CR) והזנת שורה (LF) שהוכנסו למחרוזת, כדי למנוע בעיות בהדפסה. משפט Mid שהופעל בתוכנית החליף תווים אלו ברווחים (פירוט קוד 9.6).

תוכנית 9.6: REPLACE.VBP - שימוש במשפט Mid לסילוק תווים מסוימים ממחרוזת

```
'Replace Line feeds with spaces
    nFindPos = 0
    Do
      nFindPos = InStr(sInput;Chr$(10))
      If nFindPos > 0 Then Mid(sInput, nFindPos) = " "
    Loop Until nFindPos = 0

'Replace Carriage returns with spaces
    nFindPos = 0
    Do
      nFindPos = InStr(sInput, Chr$(13))
      If nFindPos > 0 Then Mid(sInput, nFindPos) = " "
    Loop Until nFindPos = 0
```

הקוד בתוכנית 9.6 משנה את מחרוזת המקור. כלומר, הוא משנה למעשה תווים במחרוזת sInput. כיום נכללת ב- Visual Basic פונקציה חדשה, Replace, המחזירה מחרוזת חדשה עם התווים המוחלפים. התחביר הוא:

```
outString = Replace(inString, searchString, replacewith[, start[, count _
  [, compare]]])
```

פונקציה זו פועלת כשילוב של משפט Mid ופונקציית Instr. אתה פשוט מעביר את המחרוזת, את התווים שאותם בכוונתך לחפש ואת התווים המחליפים. בשימוש בפונקציה Replace, ניתן לשכתב את הקוד בתוכנית 9.6 כדלהלן:

```
'Replace Line feeds with spaces
sInput = Replace(sInput, vbCr, " ")
'Replace Carriage returns with spaces
sInput = Replace(sInput, vbLf, " ")
```

שימוש אפשרי לפונקציה Replace יכול להיות סילוק מרכאות משדה קליטת טקסט, לפני הצבתו בבסיס נתונים.

עבודה עם תווים מסוימים

מחרוזת בנויה מתווים בודדים. לכל תו כזה יש קוד מספרי, המוכר כקוד ASCII. אפשר להשתמש בקוד ASCII בפעולות נומריות על תווים, וניתן גם להשתמש בו להוספת תווים שאינם ניתנים להדפסה למחרוזת, כמודגם בתוכנית 9.6. הפונקציה Chr משמשת להחזרת תו המתאים לקוד ASCII מסוים. לדוגמה, שני המשפטים הבאים מדפיסים את המילה HELLO:

```
Print "HELLO"
```

```
Print Chr$(65) & Chr$(69) & Chr$(76) & Chr$(76) & Chr$(79)
```

הערה:

כמו פונקציות אחרות שהוזכרו לפני כן בפרק זה, Chr מופיעה בשתי צורות: Chr() מחזירה משתנה, Chr$ מחזירה מחרוזת.

בדוגמה הקודמת, הקלדת HELLO פשוטה יותר. אולם, ישנם תווים, למשל תווי הזנת שורה או הורדת גררה, שאינם ניתנים להקלדה ישירה. חייבים אז להשתמש בפונקציית Chr, באופן הבא:

```
Print "Line 1" & Chr$(13) & Chr$(10) & "Line 2"
```

הצירוף הורדת/הזנת שורה ראוי לתשומת לב מיוחדת. לעיתים קרובות יש צורך להשתמש בו כדי לאלץ שורה חדשה במחרוזת, בתיבת טקסט או בתיבת הודעה. לשם כך הוכלל ב- Visual Basic קבוע פנימי vbCrLf, כך שניתן לרשום את הקוד הקודם גם בצורה הבאה:

```
Print "Line 1" & vbCrLf & "Line 2"
```

ניתן גם להשתמש בפונקציה Chr להכללת מרכאות:

```
Print Chr$(34) & "This will be printed with quotes" & Chr$(34)
Print "This will not have quotes"
```

ניתן גם להשתמש בפונקציה Chr להחזרת כל אות ומספר. טבלה 9.3 מציגה כמה קודי ASCII שמשתמשים בהם לעיתים קרובות:

טבלה 9.3: קודי ASCII לכמה תווים נפוצים

קוד	מייצג
8	Backspace
9	טבולטור
10	הזנת שורה (LF)
13	רד שורה (CR)
32	רווח

קוד	מייצג
34	מרכאות כפולות
48	0 (הספרה אפס)
65	A
97	a

Asc היא פונקציה נלווית ל-Chr. Asc מחזירה את קוד ASCII המתאים לתו הקלט. הקוד הבא מדגים את דרך השימוש ב-Asc.

```
Print Asc("A")
```

שורה זו מחזירה את המספר 65.

מחרוזות ומספרים

ערך מחרוזת יכול למעשה לכלול ספרות נומריות כמוצג כאן:

```
Dim sStringVal As String
Dim nNumber As Integer
nNumber = 99   'A number that can be used with math expressions
sStringVal = "16 candles"  'A string that cannot be used with math expressions
```

כידוע, מספרים מסוימים מטופלים לעיתים קרובות כמחרוזות תווים. מספרי מיקוד ומספרי טלפון הן שתי דוגמאות נפוצות. לכן, לעיתים יש צורך בהסבת מספר למשתנה מחרוזת או לפונקציית מחרוזת, או להדפיסו בשילוב עם מחרוזת אחרת. באופן דומה דרוש לפעמים להשתמש במספרים המופיעים במחרוזות במשוואה מתמטית או בפונקציה נומרית.

Visual Basic מספקת את הפונקציה Str, להסבת מספר למחרוזת, ואת הפונקציה Val, להסבת מחרוזת למספר. להלן דוגמה להסבת מספר למחרוזת:

```
numstr = Str(inptnum)
```

numstr מייצג משתנה מחרוזת המכיל את פלט הפונקציה. inptnum מייצג את המספר העובר הסבה. זה יכול להיות מספר, משתנה נומרי, או פונקציה נומרית. אם inptnum הוא מספר חיובי, Str מחזירה רווח בראש המספר, כי Str שומרת מקום לתו נוסף, להצבת סימן מינוס אם יידרש.

להסבת מחרוזת למספר, ניתן להשתמש בפונקציה Val, כדלהלן:

```
numvar = Val(inptstr)
```

numvar מייצגת משתנה נומרי, המאחסן את פלט הפונקציה. inptstr יכול להיות מחרוזת תווים, משתנה מחרוזת, או פונקציית מחרוזת. הפונקציה Val מסלקת ראשית את הרווחים מהמחרוזת, ואחר-כך מתחילה לקרוא את המספרים. אם התו הראשון במחרוזת איננו ספרה (או סימן מינוס), Val מחזירה את הערך 0. אחרת - Val קוראת את המחרוזת עד שהיא נתקלת בתו הלא ספרתי הראשון. בנקודה זו נפסקת הקריאה, והפונקציה מסבה את הספרות שקראה למספר.

הערה:

בבניית מחרוזת להצגה עם שרשור, ניתן במרומז להסב מספר למחרוזת:

```
Dim sOutput As String
Dim nWeight As Integer
nWeight = 145
sOutput = "The answer is " & nWeight & " pounds"
```

עיצוב תוצאות

כשמידע, למשל מספר, מאוחסן במשתנה, הוא מאוחסן במבנה הנדרש לצורך תפעולו בקוד התוכנית. אך, לעיתים קרובות יש צורך לשנות את מבנה המידע, לפני הצגתו או הדפסתו בדוח. נניח למשל, שמספרים אחדים מאוחסנים במשתנים המייצגים ערכים דולריים. אם המספרים הם תוצאה של פעולת חילוק, ייתכן שהם כוללים שברים עשרוניים ארוכים, ויש להניח כי אם יודפסו כמות שהם, לא יהיו מיושרים כמצופה.

בקטע זה נבחן פונקציות אחדות המיועדות לסייע בעיצוב מספרים ומידע אחר. במיוחד נבחן את הפונקציה המוכרת Format, ופונקציות עיצוב ספציפיות נוספות הנכללות ב- Visual Basic 6.

פונקציות עיצוב אחרות

Visual Basic כוללת כמה פונקציות עיצוב, המיועדות לעיצוב סוגים מוגדרים של מידע. פונקציות אלו רשומות בטבלה *9.4*.

טבלה 9.4: פונקציות עיצוב חדשות שהוכנסו ל- Visual Basic 6

פונקציה	משמש ל	פלט	קלט אפשרי
FormatCurrency	כספים	8675.309	$8,675.31
FormatNumber	מספרים	-5000	(5,000.00)
FormatPercent	אחוזים	0.1234	12.34%
FormatDateTime	תאריך/שעה	"31/12 13:34"	31/12/98 1:34:00 PM
Round	מספרים	123.6	124

כל הפונקציות בטבלה 9.4 מחזירות ערכי מחרוזת. יש להתייחס אליהן כאל ביטויים (שייתכן שהם מכילים ערכי מחרוזת), וכמה פרמטרי רשות המבקרים את הפלט.

לדוגמה, FormatCurrency מעצבת את המספר בסגנון מטבע (כספים). ניתן לבחון את פונקציית FormatCurrency על ידי פתיחת החלון מיידי (Immediate Window), (אין צורך בהרצת פרויקט במחשב).

הקלדת השורה הבאה:

```
Print FormatCurrency(1234)
```

תחזיר לאחר הקלדת Enter את הפלט הבא:

```
$1,234.00
```

שים לב לסימן ($), מפריד הפסיק (,) ומספר הספרות מעבר למפריד העשרוני, המתאים להצגת ערכים כספיים. ניתן לשלוט בעיצוב באמצעות הכרטיסיה **מטבע** (Currency) ביישומון **הגדרות אזוריות** (Regional Settings) ש**בלוח הבקרה** (Control Panel).

פרמטרי רשות

לכל הפונקציות בטבלה 9.4 (למעט FormatDateTime), קיימים פרמטרי רשות והם:

- NumDigitsAfterDecimal - קובע מספר ספרות בפלט לאחר הנקודה העשרונית.
- IncludeLeadingDigit - מוסיף או מבטל את האפס המוביל בפלט שברים עשרוניים (למשל 0.567 או567.).
- UseParensForNegativeNumbers - קובע אם מספרים שליליים יוחזרו עם סימן מינוס לפניהם, או מוקפים בסוגריים (חשוב במיוחד בהנהלת חשבונות).
- GroupDigits - קובע אם סימן הפרדה בין קבוצות (פסיק מפריד) יופיע במספר המוחזר (למשל 1,234 או 1234).

רבים מפרמטרי הרשות, כמו למשל IncludeLeadingDigits, מוכרים כפרמטרים **תלת-מצביים**. לפרמטר כזה שלושה מצבים אפשריים: vbTrue (נכון), vbFalse (לא נכון), ו-vbUseDefault (השתמש בברירת המחדל). למעשה, קיימים רק שני ערכים בעלי תוקף - **נכון** ו**לא נכון**. במצב של **השתמש בברירת המחדל** הפונקציה בוחרת נכון/לא נכון לפי מה שנקבע ביישומון Regional Settings שב- Control Panel.

פונקציות חדשות לעיצוב תאריך

פרמטרי הרשות אינם קיימים ב-FormatDateTime. פונקציה זו פשוט מקבלת את הביטוי המייצג תאריך או שעה, וקבוע המייצג אחד ממבני התאריך הנקובים. מבנים אלה מופיעים בהמשך הפרק בטבלה 9.7. Visual Basic מציגה רשימת הקבועים בעת הקלדת הפונקציה בחלון הקוד. למשל, הקלד את השורה הבאה בחלון Immediate:

```
Print FormatDateTime("20:10",vbLongTime)
```

הפונקציה FormatTimeDate ממירה את השעה הנתונה 20:10, הנתונה במבנה עשרים וארבע שעות ("זמן צבאי"), למבנה 12 שעות (מבנה AM/PM) או לשעה 8:10PM. שים לב כי משתנה מחרוזת משמש כפרמטר ראשון. הביטוי חייב להיות במבנה ש-FormatDateTime מסוגלת להמיר לתאריך/שעה, אחרת תוחזר הודעת שגיאה.

שתי פונקציות נוספות שנוספו ל- Visual Basic version 6 מאפשרות שליפת השמות המודפסים של החודש או היום בשבוע מכל ביטוי תאריכי. פונקציות אלו הן

MonthName ו-WeekDayName בהתאמה. שתיהן מקבלות ערכים נומריים ומחזירות מחרוזת. פרמטרי רשות קובעים אם להשתמש בקיצורים:

```
Print MonthName(12)
```

מחזירה את המחרוזת 'December'.

```
Print MonthName(12,True)
```

מחזירה את המחרוזת 'Dec'.

גם לפונקציה WeekDayName יש פרמטרי רשות, המאפשרים לקבוע את היום הראשון בשבוע:

```
Print WeekDayName (1)
```

מחזירה את המחרוזת 'Sunday' רק אם יום זה נקבע כיום הראשון של השבוע ב- Control Panel.

```
Print WeekDayName (1, False, vbSunday)
```

מחזירה תמיד את המחרוזת 'Sunday'.

פונקציות אלו מקבלות רק ערכים נומריים. למציאת היום בשבוע או החודש בתאריך מסוים, ראשית יש להשתמש בפונקציות Visual Basic אחרות כדי לקבוע את ערכו המספרי של התאריך. לדוגמה: נניח שרוצים לדעת את היום בשבוע שבו חל 18 בפברואר 1991. ניתן להפעיל את הפונקציה WeekDay למציאת הערך המספרי של היום, ולהעבירו לפונקציה WeekDayName:

```
Print WeekDayName(WeekDay(#2/18/91#))
```

הערה:

יש לציין את התאריך על ידי # בשני קצוות התאריך. ב- Visual Basic הסולמית מציינת כי מדובר בתאריך. ניתן גם לסמן במרכאות.

עיגול מספרים

פונקציה חדשה - Round, מאפשרת עיגול ביטויים מספריים למספר מקומות נתון לאחר הסימן העשרוני. לפונקציה שני פרמטרים: הביטוי, ומספר הספרות אחרי הסימן העשרוני, אליו הוא מעוגל. נסה את הפונקציה על ידי הקלדת הביטויים הבאים בחלון Immediate:

```
Print Round(202.5)
```

מחזירה את הערך 202.

```
Print Round(202.56)
```

מחזירה את הערך 203.

```
Print Round(202.56,1)
```

מחזירה את הערך 202.6.

```
Print Round (202.56,2)
```

מחזירה את הערך 202.56.

כפי שרואים, הפונקציה Round קלה לשימוש. יש לזכור רק שהתוצאה תדרוש עיצוב נוסף אם רוצים תמיד להציב ספרות לאחר הסימן העשרוני. במילים אחרות, בביצוע שורת הקוד הבאה, לא תוצבנה ספרות לאחר הסימון העשרוני:

```
Print Round(202,4)
```

הערך המוחזר יהיה 202, ללא אפסים אחרי הסימן העשרוני.

השימוש בפונקציה Format

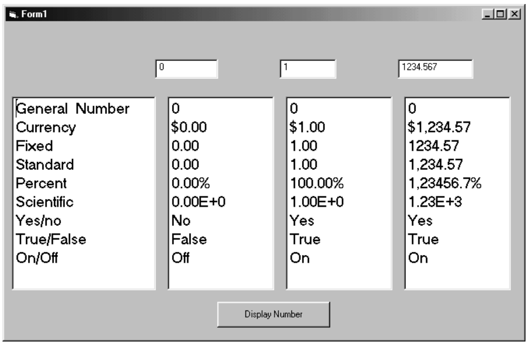
הפונקציה Format רב תכליתית יותר מאלו שנדונו עד כה. היא מסוגלת לטפל ולהחזיר תאריכים, מספרים ומחרוזות. במקום להתבסס על צרור פרמטרי רשות, משתמשת Format במחרוזת בקרה, המייצגת מבנה שנקבע על ידי המערכת או על ידי המשתמש.

עיצוב מספרים

השימוש הנפוץ ביותר בפונקציה Format הוא בהצבת מספרים במבנה מסוים, לשיפור הקריאות שלהם על ידי המשתמש. בשימוש בפונקציה Format מציין המשתמש את ערך הפלט ואת עיצובו, כמודגם בשורות הקוד הבאות:

```
Printer.Print Format(GrossSales, "Currency")
Printer.Print Format(GrossSales, "$####.00")
```

לצורך עבודה עם מספרים, ניתן להשתמש במספר תבניות בעלות שם. בכל המקרים יש צורך להקיף חלק מהתבנית במרכאות. טבלה 9.5 מראה את תבניות אלו הזמינות לפונקציה Format. תרשים 9.6 מראה איך כל תבנית כזו מציגה מספרים שונים.



תרשים 9.6: באמצעות הפונקציה Format אפשר להציג מספרים במיגוון דרכים

טבלה 9.5: תבניות נקובות בשם לפישוט הצגת מספרים

תבנית נקובה בשם	תיאור
General Number	מדפיסה את המספר ללא עיצוב מיוחד
Currency	מדפיסה את המספר עם מפריד אלפים ועם שתי ספרות לאחר הסימן העשרוני + סימן מטבע (לפי לוח הבקרה)
Fixed	מדפיסה לפחות ספרה אחת משמאל לסימן, ושתי ספרות מימינו
Standard	מדפיסה את המספר עם מפריד אלפים וכן מדפיס לפחות ספרה אחת משמאל לסימן העשרוני ושתי ספרות לימינו
Percent	מכפילה את המספר פי מאה, ומדפיסה אותו עם סימן % לימינו + 2 ספרות מימין לנקודה העשרונית
Scientific	מציגה את המספר בתבנית מספר מדעי
Yes/No	מציגה Yes לערך שונה מאפס ו-No לאפס
True/False	מציגה True לערך שונה מאפס ו-False לאפס
On/Off	מציגה On לערך שונה מאפס ו-Off לאפס

אם תבניות אלו אינן עונות לצרכיך, ניתן לעצב את התבנית המבוקשת. לצורך כך יש לקבוע היכן תמוקמנה ספרות המספר, האם ייעשה שימוש במפרידי אלפים ובסימן עשרוני, וכן לציין תווים מיוחדים שיודפסו. שורת הקוד הבאה תדפיס מספר עם ארבע ספרות אחרי המפריד העשרוני, ועם מפריד אלפים:

```
Printer.Print Format(TotalDistance, "##,##0.0000")
```

טבלה 9.6: קודים לקביעת מבנים נומריים

סימן	מטרה	תיאור
0	ממקם ספרה	מציג ספרה או מציג 0 אם אין ספרה במקום הנתון
#	ממקם ספרה	מציג ספרה או אינו מציג דבר אם אין ספרה במקום הנתון. גורם להשמטת אפסים מובילים או עוקבים.
.	מפריד עשרוני	מציין את מיקום הנקודה העשרונית
,	מפריד אלפים	מציין את מיקום מפרידי האלפים
%	סימן אחוזים	מציין את מיקום סימן האחוזים. כמו-כן גורם למספר להיות מוכפל פי מאה.
E-, E+, e-, e+	תבנית מספר מדעי	-E או -e מציג סימן מינוס למעריך שלילי, אינו מציב סימן למעריך חיובי. +E או +e תמיד מציב סימן

עיצוב תאריכים

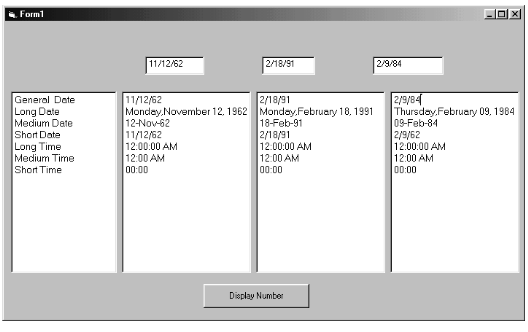
צורת נתונים נוספת הדורשת טיפול מיוחד לצורך עיצוב, הם תאריכים. הפקודה Print תגרום להצגת התאריך בתבנית ברירת המחדל של המערכת, משהו כמו 7/5/98. להצגת התאריך בתבנית אחרת (למשל July 5, 1998), יש צורך להשתמש בפונקציה Format. בטבלה 9.7 מוצגות תבניות אחדות לתאריך ולשעה (ראה תרשים 9.7)

טבלה 9.7: תבניות נקובות בשם לתאריך ולשעה

תבנית	תיאור
General Date	מציגה את התאריך והזמן אם הביטוי כולל את שניהם. אחרת מציגה תאריך או שעה
Long Date	מדפיסה את היום בשבוע, היום בחודש, החודש, השנה
Medium Date	מדפיסה את היום בחודש, קיצור בן שלוש אותיות לשם החודש ואת השנה
Short Date	מדפיסה את היום, החודש והשנה - למשל 7/5/98
Long Time	מדפיסה שעה, דקות ושניות, ואת הסימן AM/PM
Medium Time	מדפיסה שעה ודקות, ואת הסימן AM/PM
Short Time	מדפיסה שעה ודקות בתבנית 24 שעות (זמן צבאי)

בנוסף לתבניות המופיעות בטבלה 9.7, קיימים קודים המאפשרים למשתמש ליצור תבנית תאריך ושעה. למשל, שורת הקוד הבאה תדפיס את תאריך היום בצורה:

```
Form1.print Format(Now, "mm/dd/yyyy")
```



תרשים 9.7: שימוש בתבניות מיוחדות משפר את המראה של מידע תאריך

הפונקציה Format משתמשת בפלט הפונקציה Now, ומדפיסה רק את החלק המצויג בקודי האותיות. רשימה מלאה של קודים אלה מופיעה במערכת Help תחת הנושא **User-defined test formats**.

עבודה עם ערכי תאריך

ראינו כיצד ניתן לעצב תאריכים. Visual Basic כוללת גם פונקציות אחדות המאפשרות עבודה עם תאריכים כערכים, בדומה לעבודה עם מספרים במשוואות. זה אפשרי כי סוג הנתונים Date, הקיים ב- Visual Basic, מאחסן תאריך (ושעה) במבנה הניתן לשימוש בפונקציות שונות. הפונקציה Format, כמו פונקציות רבות אחרות מקבלת פרמטר תאריך מטיפוס String או Date. אך יש פונקציות המקבלות רק משתנים שהם תאריכים. להסבת מספר לתאריך, ניתן להשתמש בפונקציה CDate. יש להיזהר, כי מחרוזות בלתי מתאימות תגרומנה לשגיאה:

```
Print CDate("November 23, 1963")
```

מחזירה את התוצאה 11/23/63,

```
Print CDate("oops")
```

מחזירה הודעת שגיאה.

לבדיקה אם ניתן להסב מחרוזת לתאריך, משמשת הפונקציה IsDate. בשתי שורות הקוד שלעיל תחזיר פונקציה זו True ו-False בהתאמה.

חילוץ חלקי תאריך

במקרים שבהם נדרש לעבוד בתאריכים בדרך אוטומטית, יש לפעמים צורך בשליפת חלק מתאריך. נניח למשל שיש צורך לבצע תהליך מסוים בכל יום רביעי בשבוע. בקטעים הקודמים הודגם איך ניתן לקבוע את מספר היום בשבוע, בעזרת הפונקציה WeekDay, ואת התאריך והשעה הנוכחית בעזרת Now. כמו משימות רבות ב- Visual Basic, גם זו ניתנת לביצוע בדרכים אחדות. ניתן למשל להשתמש בפונקציה Format לשליפת מספר היום בשבוע כדלהלן:

```
If Format(Now, "w") = 4 Then RunTheProcess
```

Visual Basic כוללת גם פונקציות לשליפת היום, החודש והשנה. אלו נקראות כמובן Day, Month ו-Year.

עבודה עם מרווחי תאריך

שתי פונקציות נוספות השימושיות לעבודה עם תאריכים, הן DateAdd ו-DateDiff. ניתן להשתמש ב-DateAdd להוספת כל סוג של מרווח זמן לתאריך, כשהתוצאה המוחזרת היא התאריך החדש. לדוגמה, במקרה שנדרש להכין היום דוח, שהתאריך שלו הוא יום אתמול, ניתן להשתמש ב-DateAdd באופן הבא:

```
Dim dtYesterday As Date
dtYesterday = DateAdd("d",-1,Now)
```

הקוד מורה בפשטות לפונקציה DateAdd להוסיף מרווח של מינוס יום אחד לתאריך המערכת הנוכחי. קודים המתאימים (d, w, וכדומה), הנכללים בפונקציה Format, יכולים לשמש לקביעת סוג המרווח.

בעוד ש-DateAdd מאפשרת הוספת מרווח לתאריך, הפונקציה DateDiff מאפשרת חישוב המרווח בין שני תאריכים. מאפיין זה שימושי בתוכניות הקובעות לוח-זמנים, למשל הקוד הבא, המאפשר הרצת תהליך אחת לשעה:

```
Dim dtLastRun As Date
Dim dtNextRun As Date
Dim nMinRemain As Integer

dtLastRun = Now
dtNextRun = DateAdd("n", 60, Now)

nMinRemain = DateDiff("n", Now, dtNextRun)
While nMinRemain > 0
	Debug.Print "Start in " & nMinRemain & " minutes."
	NMinRemain = DateDiff("n", Now, dtNextrun)
	DoEvents
Wend

MsgBox "OK Done waiting!"
```

הלולאה While בדוגמת הקוד שלעיל תרוץ כל זמן שהמרווח שבין הזמן הנוכחי למועד הפעולה הבאה גדול מאפס.

שנת 2000

רבות נכתב לאחרונה על "בעיית שנת 2000", ועל האנדרלמוסיה שהיא עלולה לגרום במערכות ממוחשבות. הבעיה היא שמחשבים (ואנשים) רבים משתמשים בשתי ספרות לציון השנה, מה שעלול לגרום לבלבול. האם 2/8/00 משמעותו 2/8/1900 או 2/8/2000? לבעיה זו אין פתרון פשוט. קשה להניח ש-2/8/00 מתייחס לשני באוגוסט 2000, אם הוא מייצג את תאריך הולדתו של אדם בן מאה שנה.

בנוסף, לתוכניות רבות יש נתונים וחישובים המבוססים על תאריכי שנה דו-ספרתיים. נתייחס לשורות הבאות מתוך טבלה של בסיס נתונים:

תאריך	כמות
960101	123.45
960102	432.34
.	
.	
991231	442.00

בתיק זה תאריכים מאוחסנים כמספרים שלמים, מה שמקל על איתור טווח תאריכים נתון. למשל: כדי לסכם את כל הכמויות משנת 1997 והלאה, בחר את כל השורות

שבהן התאריך שווה או גדול מ-970101. אולם נניח שהתאריך הראשון בשנת 2000 מוצג כ-00010. במקרה זה התוכנית תיכשל.

למזלנו, בעולם המחשבים האישיים ניתן להתגבר בקלות יחסית על בעיות מסוג זה. הבעיות ב- Visual Basic תהיינה לא עם תאריכים המאוחסנים בבסיסי נתונים, אלא עם נתונים המוזרמים בידי משתמשים, שבמקרים מסוימים מעדיפים את שיטת השנה הדו-ספרתית.

נתייחס לקוד הבא:

```
Dim dt As Date
dt = "01/01/00"
Debug.Print "Date is " & FormatDateTime(dt, vbLongDate)
```

התאריך המאוחסן במשתנה שייך לשנת 2000. מבחינת המפתח, חייבת להתקבל החלטה אם ראוי להמשיך להשתמש לציון השנה בסימון דו-ספרתי. אם כן, התוכנית חייבת לכלול לוגיקה מתאימה להסבת הסימן הדו-ספרתי לשנה הנכונה. במקרים מסוימים סימון שנה דו-ספרתי עלול להיות בלתי קביל, ומשתמשים חייבים ללמוד לקבל עובדה זו.

בכל מקרה, עם קצת תכנון והוספת לוגיקה מתאימה בתוכניות, יוכל כל משתמש להצהיר בגאווה, כי תוכניות Visual Basic שלו מוכנות לשנת 2000.

מכאן...

פרק זה מספק מבט כללי על התכנות ב- Visual Basic. נלמדו בו העקרונות של בניית משפטי קוד וכן דרך הטיפול והעיצוב של סוגי מידע שונים. לחומר נוסף על תכנות יש לפנות לפרקים הבאים:

- ללמוד על בקרת הזרימה של קוד התוכנית ועל קבלת החלטות - פרק 10 **שליטה במהלך התוכנית**.
- ללמוד על בניית פונקציות - פרק 11 **ניהול הפרויקט: תת-שגרות, פונקציות וריבוי טפסים**.
- ללמוד כיצד לכתוב ממשקים מקצועיים - פרק 18 **תכנון ממשק נכון**.

10

שליטה במהלך התוכנית

מה בפרק?

- קבלת החלטות בתוכנית
- עבודה עם לולאות
- ניפוי שגיאות
- איתור שגיאות

בפרק 9 **יסודות התכנות של Visual Basic**, למדת כיצד ליצור מספר פקודות קוד. כעת, תוכל להרחיב את הידע שצברת על ידי שליטה בסדר הביצוע של חלקי הקוד שלך.

שתי היכולות העיקריות של מחשבים הן ביצוע הוראות במהירות וקבלת החלטות בדייקנות. בפרק זה, תלמד כיצד לנצל יכולות אלו בשני מישורים עיקריים: יישום שיטות שונות לקבלת החלטות ושימוש בלולאות, המאפשרות ביצוע חוזר של חלקים מסוימים מתוך הקוד. כמו כן, תלמד כיצד לבנות קוד המגיב לשגיאות העלולות להתרחש בעת פעולת התוכנית.

קבלת החלטות בתוכנית

בפרק 9 ערכת היכרות עם מחלקה של פקודות קוד, המוכרות בשם **פקודות משימה** (Assignment Statements), המשמשות להגדרה ושינוי ערכי משתנים. קבוצה חשובה נוספת של פקודות קוד, המיועדת לטיפול במשימות מורכבות יותר, מכונות **פקודות שליטה** (Control Statements). ללא פקודות שליטה, יבוצע הקוד מתחילתו ועד סופו, שורה אחר שורה ללא הבחנה, והתוכנית תיעצר לאחר ביצוע שורת הקוד האחרונה.

סוג אחד של פקודות המקנות שליטה על אופן ביצוע הקוד הן **פקודות החלטה** (Decision Statements), השולטות על יישום חלקים שונים בתוכנית, בהתאם לתנאים המתקיימים בזמן ביצוען. שני מבני ההחלטה הנפוצים בתוכניות Visual Basic מבוססים על פקודות If ופקודות Select Case, בהן יעסקו הסעיפים הבאים.

שימוש בפקודה If

במקרים רבים, הנך מעוניין בביצוע שורה אחת או מספר שורות קוד רק אם תנאי מסוים מתקיים, כלומר מקבל את הערך True. קיימות שתי צורות לטיפול בהתניות True באמצעות הפקודה If: הצורה הראשונה היא שימוש בפקודת If בת **שורה אחת** והצורה השנייה היא שימוש בפקודת If בעלת **מספר שורות**. בכל אחת מהצורות נעשה שימוש בפקודה If לבדיקת קיום תנאי מסוים. אם הערך המתקבל הוא True, התוכנית תבצע את הפקודות המצורפות לפקודה If. לדוגמה, שתי פקודות If הבאות מבצעות פעולה דומה:

```
' Single line IF
If x > 5 Then x = 0
```

```
' Multiple line IF
If x > 5 Then
	x = 0
End If
```

שימוש בשורות מפוצלות ומורחקות מן השוליים הוא מנהג מקובל המקל על קריאת הקוד בעת שימוש במשפטי If מרובי שורות.

אם הערך המתקבל הוא False (בדוגמה לעיל, אם X **אינו** גדול מ-5) התוכנית מדלגת על שורת If (במשפטי If בעלי שורה אחת) או על השורות שבין הפקודות If ו-End If (במשפטי If בעלי מספר שורות) ועוברת לביצוע השורה הבאה.

פקודות If בנות שורה אחת

פקודות If של שורה אחת גורמות לתוכנית לבצע משימה בודדת אם הערך המתקבל הוא True. משימה זו יכולה להיות פקודה בודדת, או מספר פקודות הנכללות בשיגרה. להלן תחביר פקודת If בעלת שורה בודדת:

```
If Condition Then command
```

הארגומנט *Condition* מייצג תנאי לוגי כלשהו היכול להשתייך לאחת הקבוצות הבאות:

- השוואה של משתנה ל**אותיות** (Literal), משתנה אחר או פונקציה.
- משתנה או בסיס נתונים המכיל ערך של True או False.
- כל פונקציה או ביטוי המחזיר ערך של True או False.

הארגומנט *command* מייצג את המשימה שתבוצע אם התקבל הערך True. משימה זו יכולה לכלול כל פקודה חוקית ב- Visual Basic, כולל קריאה לשיגרה. הקוד הבא מדפיס שם עובד בתנאי שמשכורתו גבוהה מערך מקסימלי קבוע מראש:

```
If cSalary > cMaxSalary Then printer.print sEmpName
```

פקודות If מרובות שורות

אם ברצונך לבצע יותר מפקודה בודדת בתגובה להתניה, תוכל להשתמש בפקודת If מרובת שורות, המוכרת גם כ**בלוק If** (Block If). מבנה זה מלכד מספר פקודות קוד בין שורת If לשורת End If. אם הערך שהתקבל הוא True, תבצע התוכנית את בלוק הפקודות שבין If ל-End If. אם הערך שהתקבל הוא False, התוכנית תדלג ישירות לשורה שאחרי End If. הדוגמה הבאה מראה כיצד ניתן להשתמש ב**בלוק If** לחישוב התשלום עבור השתתפות בתערוכה. אם הופקדה מקדמה מראש, היא מצורפת לתשלום הכללי והתוכנית קוראת לשיגרה המטפלת בהזמנה:

```
If cDepositAmt > 0 Then
      cTotalPaid = cTotalPaid + cDepositAmt
      cDepositAmt = 0
      Call UpdateReservation
End If
```

שימוש בהתניית False

כמובן שאם אפשרית התניית True, אזי אפשרית גם התניית False. במקרים מסוימים, תרצה שתוכניתך תבצע משימה מסוימת רק כאשר מתקבל הערך False. במקרים אחרים תהא מעוניין לבצע משימות שונות בהתאם לערך המתקבל (True או False). הסעיף הבא מתייחס לחלק ההתניה המטפל בערך False.

שימוש באופרטור NOT

דרך אחת ליצור פקודה או קבוצת משפטים המטפלים בערך false, היא שימוש באופרטור Not. האופרטור Not למעשה הופך את התנאי הבא אחריו. כלומר, אם ההתניה כשלעצמה היא בעלת ערך True, הופך האופרטור את ערך הביטוי כולו ל-False ולהפך. הקוד הבא משתמש באופרטור Not להיפוך המשתנה הבוליאני bTaxable, המדווח על כמות המכירות של המציג בתערוכה תוך שימוש **במדרגת מס** (Tax Status). bTaxable מקבל ערך True אם המציג בתערוכה צריך לשלם מס על מכירות ושימוש, וערך False אם לא. הקוד עורך בדיקה על הערך המתקבל מן הביטוי Not bTaxable. אם הערך המתקבל הוא True, אזי המציג בתערוכה פטור ממס.

```
If Not bTaxable Then
      cSalesTax = 0
      cUseTax = 0
End If
```

שימוש בהתניות True ו-False

דרך נוספת לטיפול בהתניות False היא הגדרת מערכות שונות של פקודות אשר תבוצענה בהתאם לערך המתקבל (True או False). תוכל להסתייע לשם כך בביטוי Else, אותו ניתן לשלב כחלק מבלוק If.

לטיפול, הן בהתניית True והן בהתניית False, השתמש בבלוק If ושלב בו את הביטוי Else בצורה הבאה:

```
If Condition Then
      statements to process when Condition is True
Else
      Statements to process when Condition is False
End If
```

פקודות If ו-End If בבלוק לעיל, זהות לדוגמה הקודמת. גם כאן ההתניה היא ביטוי לוגי המניב ערך True או False. המרכיב המרכזי במערכת משפטים זאת הוא הביטוי Else. ביטוי זה משולב אחרי המשימה האחרונה לביצוע אם מתקבל הערך True, ולפני המשימה הראשונה לביצוע אם מתקבל הערך False. אם ערך ההתניה הוא True, התוכנית תבצע את המשימות המופיעות **לפני** הביטוי Else ומשם תפנה ישירות לשורה שאחרי End If ותדלג על השורות שבין Else ו-End If. אם ערך ההתניה הוא False, התוכנית תפנה ישירות לביצוע המשימות המופיעות **אחרי** הביטוי Else.

הערה:
אם ברצונך לבצע משימה כלשהי רק אם ערך ההתניה הוא False, תוכל לכתוב קטע קוד רק לחלק שבין הפקודות Else ו- End If. אינך חייב לכתוב קוד גם בחלק שבין הפקודות If ו-Else.

טיפ:
כאשר מופיעים מספר משפטים בין פקודות If ו-End If מומלץ לעיתים לחזור על ההתניה גם בפקודה End If, כדי להקל על קריאת הקוד, כמו בדוגמה הבאה:

```
If cTotalSales > cProjectedSales Then
	קבוצה של משפטי קוד
Else
	קבוצה נוספת של משפטי קוד
End If
```

שימוש בהתניות מרובות

בסעיפים הקודמים ערכת היכרות עם בלוק If פשוט הכולל התניה בודדת, ומסוגל לבצע משימות עבור ערך True או False. ביכולתך לכלול מספר התניות בבלוק If אחד. פקודת ElseIf מאפשרת להוסיף התניה נוספת, אליה מתייחסת התוכנית אם ההתניה הקודמת מקבלת ערך False. הפקודה ElseIf מאפשרת לשלב מספר בלתי מוגבל של התניות בבלוק If יחיד. להלן דוגמה לשימוש בפקודה ElseIf לבדיקת המשתנה fTest, המסוגל להכיל שלושה ערכים אפשריים: מספר שלילי, אפס, או מספר חיובי.

```
If fTest < 0 Then
	lblResult.Caption = "Negative"
ElseIf fTest = 0 Then
	lblResult.Caption = "Zero"
Else
	lblResult.Caption = "Positive"
End If
```

תחילה בודקת התוכנית את ההתניה של פקודת If. אם הערך המתקבל הוא True, מבצעת התוכנית את המשימה (או המשימות) המצורפת לפקודת If, ולאחר מכן מדלגת ישירות לשורה הבאה אחרי פקודת End If.

אם הערך המתקבל מההתניה הראשונה הוא False, מדלגת התוכנית לפקודה ElseIf המופיעה ראשונה ובודקת את ההתניה שלה. אם הערך המתקבל הוא True, מבצעת התוכנית את המשימה המצורפת לפקודה ElseIf ושוב מדלגת אל השורה הבאה אחרי פקודת End If. התהליך מתבצע באופן דומה על כל משפטי ElseIf המשולבים בבלוק.

אם עבור כל ההתניות התקבל הערך False, מדלגת התוכנית לפקודה Else ומבצעת את המשימות המופיעות בינה לבין פקודת End If. פקודת Else היא אופציונלית ואין חובה לכלול אותה בבלוק.

שימוש בפקודה Select Case

דרך נוספת לקבלת החלטות בתוכנית היא שימוש בפקודה Select Case, המאפשרת לך לבצע שורת משימות בהתאם לערך המתקבל עבור ביטוי מסוים, בין אם הוא משתנה בודד או ביטוי מורכב. פקודת Select Case מורכבת משני חלקים: הביטוי עליו מופעלת הפקודה ושורת משפטי Case המבטאים ערכים אפשריים עבור הביטוי ומשימות אשר על התוכנית לבצע אם ערכים אלה מתקבלים.

כיצד עובדת הפקודה Select Case

מבנה ההחלטה Select Case דומה למבנה ההחלטה If\Then\ElseIf. שורות הקוד הבאות מדגימות את צורת הבנייה של בלוק Select Case:

```
Select Case testvalue
	Case Value 1
		Statement group 1
	Case Value 2
		Statement group 2
End Select
```

המרכיב הראשון בבלוק היא פקודת Select Case. פקודה זו מגדירה את הביטוי המוערך מול מספר תוצאות אפשריות. הביטוי המוערך, אשר מיוצג בדוגמה על ידי הארגומנט testvalue, יכול להיות מספר או מחרוזת חוקיים, כולל **Literals**, משתנים או פונקציות.

פקודת Case נכתבת בראש כל קבוצה מותנית של פקודות (אשר תבוצענה רק אם יתקבל הערך המוגדר). פקודה זו מגדירה ביטוי (המיוצג על ידי הארגומנטים Value 1 בפקודה הראשונה ו- Value 2 בפקודה השנייה) המושווה ל-TestValue. אם שני הביטויים שווים, התוכנית תבצע את המשימות המקושרות לפקודה Case. כלומר המשימות המופיעות בין שתי פקודות Case או בין הפקודה Case לפקודה End Select. אם שני הביטויים אינם שווים, התוכנית ממשיכה הלאה ומשווה את TestValue מול פקודת Case הבאה.

הפקודה End Case מציינת את סוף הבלוק Select Case.

הערה:

התוכנית מבצעת קבוצת פקודות אחת בלבד עבור כל בלוק Select Case. זאת גם במקרה שיש יותר מפקודת Case אחת התואמת את ערך הביטוי המוערך.

אזהרה:

הביטויים TestValue ו-Value חייבים לייצג אותו סוג של נתונים. לדוגמה, אם TestValue הוא מספר, גם כל הערכים המושווים אליו במשפטי Case צריכים להיות מספרים.

פקודות Case המשולבות בבלוק Select Case יכולות לטפל בנוסף לערכים נפרדים, גם **ברשימות** (Lists), **תחומים** (Ranges) **והשוואות** (Comparisons) ערכים. שים לב, בדוגמה הבאה, לשימוש ב- Case Is < 0, Case 4 to 9 ו- Case Is > 50:

```
nQtyOrderd = Val(txtQuantity)
Select Case nQtyOrdered
	Case Is < 0 'note use of comparison
		MsgBox "Order quantity cannot be negative!", vbExclamation
		Exit Sub
	Case 1, 2, 3 'note use of list
		fDiscount = 0
	Case 4 to 9 'note use of range
		fDiscount = 0.03
	Case 10 to 49
		fDiscount = 0.08
	Case is > 50
		 fDiscount = 0.1
End Select
```

טיפול בערכים אחרים

הדוגמאות שהוצגו עד כה עובדות היטב כאשר הביטוי המוערך שווה לאחד הביטויים בפקודות Case. אך מה לגבי ערכים שאין אליהם התייחסות בפקודות Case? תוכל להתאים את הקוד שלך כך שיטפל גם בערכים מסוג זה על ידי שימוש בפקודה Case Else. הפקודה Case Else נכתבת לאחר המשימה המסיימת את פקודת Case האחרונה בבלוק. אחריה תיכתבנה המשימות לביצוע ואחריהן פקודת End Select.

תוכל להשתמש בפקודה Select Case לחישוב ערכים שלא הוגדרו בפקודות Case או לחילופין, כדי להסב את תשומת לב המשתמש לכך שהקליד ערך בלתי חוקי.

דמה לעצמך טופס פשוט שנועד לשיבוץ ציוני בחינה של תלמיד בשלוש תיבות דו-שיח. בטופס קיים גם לחצן המחשב את ממוצע הבחינות ומשתמש בבלוק Select Case לביטוי ציון התלמיד באמצעות אות לועזית, בהתאם לממוצע בחינותיו. תוכנית הדוגמה 10.1 מציגה את הקוד שבשגרת האירוע Click של לחצן החישוב.

תוכנית דוגמה 10.1: SELCASE.VBP שימוש ב- Select Case לביטוי ציון התלמיד באמצעות אות לועזית

```
Private Sub cmdCalc_Click()
	Dim nSum As Integer
	Dim fAvg As Single
	nSum = Val(txtTest1) + Val(txtTest2) + Val(txtTest3)
	fAvg = nSum / 3
	lblAverage = FormatNumber(fAvg, 1)
```

```
    Select Case Round(fAvg)
        Case Is = 100
            lblGrade = "A+"
        Case 93 To 99
            lblGrade = "A"
        Case 83 To 92
            lblGrade = "B"
        Case 73 To 82
            lblGrade = "C"
        Case 63 To 72
            lblGrade = "D"
        Case Is < 63
            lblGrade = "F"
    End Select
End Sub
```

בצע את הצעדים הבאים כדי לבדוק את ביצועי התוכנית:

1. הוסף לטופס שלוש תיבות טקסט בשם txtTest1, txtTest2 ו-txtTest3.
2. הוסף לחצן פקודה בשם cmdCalc.
3. כמו בכל טופס, עליך להוסיף פקדי Label לפי הצורך. במקרה זה עליך להוסיף שני פקדים להצגת התוצאות: lblAverage ו-lblGrade. הוסף את שני הפקדים לטופס והגדר את מאפיין Caption כמחרוזת ריקה.
4. הכנס את הקוד שבתוכנית הדוגמה 10.1 לשגרת Click של לחצן cmdCalc.
5. לחץ על לחצן ההפעלה של Visual Basic.

עבודה עם לולאות

סוג עיקרי נוסף של משפטים המאפשרים שליטה בשטף הקוד הן **לולאות** (Loops). אתה משתמש בלולאות לשם ביצוע חוזר של משימות בתוכניתך. שלושת סוגי הלולאות העיקריים ב- Visual Basic הם **לולאות מנייה** (Counter Loops), **לולאות תנאי** (Conditional Loops) **ולולאות מספור** (Enumeration Loops). לולאות מנייה, או לולאות...For, מבצעות משימה מספר מוגדר של פעמים. לולאות תנאי, או לולאות ...Do, מבצעות משימה כל עוד תנאי מסוים מתקיים או עד אשר תנאי מסוים מתקיים. לולאות מספור, משמשות לביצוע פעולה מסוימת על כל פריט בקבוצת אובייקטים. הסעיפים הבאים יציגו בפניך את סוגי הלולאות השונים.

לולאות For

לולאת מנייה מוכרת גם בכינוי לולאת For, או לולאת For...Next. בתחילת כל לולאת For יש להגדיר משתנה בתור מונה, ולקבוע את ערכי הפתיחה והסיום של המונה. ניתן, כאופציה, גם להגדיר ערך **דילוג** (Step) הקובע את גודל הצעדים על פיהם יגדל (או יקטן) ערך המונה, בכל פעם שהתוכנית תבצע את הלולאה. כאשר התוכנית מבצעת את הלולאה בפעם הראשונה, המונה מותאם לערך הפתיחה. מכאן והלאה, בכל פעם שהלולאה מבוצעת מחדש, ערך המונה משתנה בדילוגים קבועים, בהתאם לערך הדילוג (Step), עד אשר ערך המונה חולף על פני ערך הסיום. אז מפסיקה התוכנית את ביצוע הלולאה ומדלגת אל משפט התוכנית הבא (הנמצא אחרי פקודת Next של הלולאה).

אזהרה:

במצב רגיל, אם ערך הפתיחה של הלולאה גדול מערך הסיום, התוכנית לא תבצע את הלולאה כלל. עם זאת, ניתן לגרום למונה לצעוד אחורה (כלומר מערך פתיחה גדול לערך סיום קטן יותר) על ידי הגדרת ערך דילוג (Step) שלילי. במקרה זה הלולאה תתבצע עד אשר ערך המונה יהיה קטן מערך הסיום.

טיפ:

כדי להקל על קריאת התוכנית, מומלץ להוסיף את שם המשתנה, המשמש כמונה, גם לפקודה Next. דבר זה נחוץ בעיקר בעת עבודה עם לולאות משנה המשולבות בתוך לולאות ראשיות.

אזהרה:

באפשרותך להשתמש בכל אחד מסוגי המשתנים המספריים כמונה בלולאה. עם זאת, עליך להביא בחשבון את המגבלות של סוגי המשתנים השונים. לדוגמה, אם ברצונך לבצע לולאה כלשהי 40,000 פעמים, שימוש במשתנה מסוג Integer, כמונה, יגרום לתוכניתך להציג הודעת שגיאה, זאת משום שהערך המקסימלי שמשתנה מסוג זה יכול להכיל הוא 32,767. מגבלות המשתנים לסוגיהם השונים מתוארים בקובץ העזרה **Data Type Summery**.

הקוד הבא מראה כיצד ניתן להשתמש בלולאת For...Next כדי להדפיס את המספרים 1 עד 10 ואת שורשיהם הריבועיים.

```
For I = 1 to 10
	sTemp = I & " squared is " & (I * I)
	Printer.Print sTemp
Next I
```

הדוגמה הבאה מראה כיצד ניתן להסתייע בלולאת For לאיפוס מערך משתנים. שימוש מסוג זה בלולאות For...next לביצוע פעולות על מערכי משתנים הוא נפוץ למדי ויכול לחסוך שורות קוד רבות:

```
For I = 1 to 12
        fMonthSales(I) = 0
        fMonthExpenses(I) = 0
        fMonthProfit(I) = 0
Next I
```

הערה:

כפי שראית בדוגמה האחרונה ועוד תראה בדוגמאות אחרות בספר, מקובל למדי לשלב בין לולאות For ומערכי משתנים. לולאת For מהווה צורה נוחה לבחון או לשנות פריטים במערכי נתונים שכן מונה הלולאה יכול לשמש כאינדקס המערך.

אזהרה:

לעולם אל תאפס את ערך הפתיחה של המונה בתוך הלולאה עצמה. דבר זה יגרום ללולאה להתבצע אינסוף פעמים.

במרבית המקרים תרצה שלולאת For שלך תתבצע עבור כל הערכים האפשריים בין ערך הפתיחה לערך הסיום. עם זאת, ייתכנו מצבים בהם תרצה לנטוש את ביצוע הלולאה קודם לכן. לצורך כך עליך להוסיף ללולאה פקודת Exit For אשר תופיע בנקודה בה אתה מעוניין לעצור. פקודת Exit For בדרך כלל מקושרת לפקודה If באמצעותה בודקת התוכנית האם יש לעצור את הלולאה. קטע הקוד הבא הוא חלק ממשחק טריוויה בו נבדקת כמות הנקודות של השחקן לאחר כל שאלה. אם הניקוד יורד אל מתחת לאפס, לא תוצגנה יותר שאלות עבור השחקן.

```
For I = 1 To nNumQuestions
        Call AskAquestion
        If bPlayerWasRight Then
                nScore = nScore + nQValue
        Else
                nScore = nScore - nQvalue
        End If
        If nScore <= 0 Then Exit For
Next I
```

לולאות Do

המרכיב העיקרי בלולאת תנאי הוא, כמובן, התנאי עצמו. תנאי יכול להיות כל ביטוי המסוגל להשיב ערך True או False. תנאי יכול להיות פונקציה, כגון EOF; ערך של מאפיין, כגון מאפיין Value של לחצן פקודה; או ביטוי, כגון numval > 15. לולאות תנאי נחלקות לשני סוגים עיקריים: **לולאות Do While**, המבוצעת **כל עוד** ערך התנאי הוא True (כלומר, בשעה שהתנאי מתקיים) **ולולאת Do Until**, המבוצעת **עד אשר** ערך התנאי הוא True (כלומר הלולאה מתבצעת כל עוד ערך התנאי הוא False).

הערה:
הביטוי EOF מבטא End-Of-File. הפונקציה EOF משיבה ערך True אם הגעת לסוף קובץ או סוף Recordset (אוסף רשומות).

לולאות Do While

מילת המפתח While גורמת לתוכנית לבצע את הלולאה כל עוד ערך התנאי הוא True. כאשר ערך התנאי משתנה ל-False התוכנית מפסיקה את ביצוע הלולאה ומדלגת למשפט התוכנית המופיע אחרי פקודת Loop.

ניתן לכתוב לולאת Do While בשתי צורות שונות. ההבדל מתבטא במיקום התנאי. ניתן למקם את התנאי בתחילת או בסוף הלולאה.

הדוגמה הבאה מציגה את הצורה הראשונה, בה מופיע התנאי בתחילת הלולאה. הקוד גורם לתוכנית לבצע את הלולאה כל עוד יש רשומות זמינות ב-Recordset.

```
' This code assumes the following:
	' db is an open database with data
	' rs is an open recordset with data
	' lstdata is a listbox
	' lblcount is label
Do While Not rs.EOF
	lstData.AddItem rs.fields(0)
	lblCount = lstData.ListCount
	DoEvents
	rs.Movenext
Loop
```

הערה:
לעת עתה נניח ש-RS הינו אוסף רשומות כלומר אוסף שמות תלמידים + מספר טלפון, לדוגמה:

איל גוטליב, 03-5077011
רוני ארבל, 02-5311721

וכל רשומה מכילה מספר שדות. בדוגמה שלנו יש 2 שדות (Fields): שם + מספר טלפון.

שילוב הביטוי While בפקודה Do המופיעה בתחילת הלולאה, מורה לתוכנית לבדוק את התנאי **לפני** שהיא ניגשת לביצוע הפקודות שבגוף הלולאה. אם הבדיקה מעלה כי ערך התנאי הוא True, התוכנית מבצעת את שורת המשימות המופיעות בין הפקודות Do ו-Loop, ולאחר מכן חוזרת לפקודה Do ובודקת את התנאי מחדש. ברגע שערך התנאי משתנה ל-False, התוכנית מדלגת לשורה הכתובה לאחר פקודת Loop. המשפטים Do ו-Loop הם מרכיבים בלתי נפרדים ממבנה הלולאה.

למעשה, כאשר אתה בונה את הלולאה בצורה זו, ייתכן מצב בו המשימות בגוף הלולאה לא תבוצענה כלל. כאשר ערך התנאי הוא False מלכתחילה, התוכנית תדלג ישירות לקטע הקוד הבא, מבלי לבצע את הלולאה.

אם אתה מעוניין שהתוכנית תבצע את לולאת Do While פעם אחת לפחות, עליך להשתמש בצורת הכתיבה השנייה, בה מופיע התנאי בסוף הלולאה במקום בתחילתה. באופן זה התוכנית מבצעת את המשימות שבגוף הלולאה לפחות פעם אחת לפני בדיקת התנאי עצמו. בסוף הבדיקה מחליטה התוכנית האם לבצע את הלולאה פעם נוספת או לדלג לקטע הקוד הבא:

```
Do
      lstData.AddItem rs.Fields(0)
      lblCount = lstData.ListCount
      DoEvents
      rs.Movenext
Loop While Not rs.EOF
```

היה זהיר בעת שימוש בלולאה בצורה זאת, וקח בחשבון כי הפקודות המופיעות בין המשפטים Do ו-Loop יבוצעו לפחות פעם אחת ללא בדיקת התנאי המוגדר בלולאה. לדוגמה, אם הערך של פונקציית EOF הוא True עוד בטרם ביצוע הלולאה בפעם הראשונה, התוכנית תפיק הודעת שגיאה בעת ניסיון לגשת לרשומה הנוכחית במסגרת הלולאה.

אזהרה:

אל תוסיף את הביטוי While בפקודות Do ו-Loop בו-זמנית. פעולה זו תגרום לשגיאה בעת ניסיון להפעיל את התוכנית.

הערה:

אם אתה עובד על קוד שפותח על ידי מישהו אחר, ייתכן שתיתקל בלולאה המתחילה בפקודה While ומסתיימת בפקודה Wend (הביטוי Wend מבטא "While End", כלומר את סוף לולאת While). צורה זו, המהווה שריד מגרסאות Visual Basic ישנות יותר, עובדת באופן זהה ללולאת While, כשהתנאי נמצא בתחילת הלולאה, כחלק מפקודת Do. Visual Basic עדיין תומכת בלולאות מסוג זה, אך מומלץ להשתמש בצורה המודרנית והגמישה יותר של Do...While.

לולאות Do Until

לולאות Do Until זהות ביסודן ללולאות Do While, אלא שהתוכנית מבצעת את המשימות בגוף הלולאה כל עוד ערך התנאי הוא False. כאשר ערך התנאי משתנה ל-True הלולאה נעצרת. כמו קודמתה, גם את לולאת Do Until ניתן לכתוב בשתי צורות שונות: בראשונה התנאי הוא חלק מפקודת Do, ובשנייה התנאי הוא חלק מפקודת Loop. כאשר התנאי הוא חלק מפקודת Do, הוא נבדק בטרם ניגשת התוכנית לביצוע המשימות. כאשר התנאי מופיע בסוף הלולאה, כחלק מפקודת Loop, התוכנית מבצעת את המשימות בגוף הלולאה לפחות פעם אחת לפני בדיקת התנאי עצמו.

נהוג להשתמש בלולאות Do Until בעת קריאה ועיבוד קבצי נתונים. הלולאה מתחילה ברשומת הקובץ הראשונה ומעבדת כל רשומה עד לסוף הקובץ. שורות הקוד הבאות עושות שימוש בלולאת Do Until כדי לטעון את שמות כל המחברים שבמסד הנתונים BIBLIO.MDB (מסד נתונים לדוגמה המצורף ל- Visual Basic) לתוך תיבת רשימה:

```
Dim dbTest As Database
Dim rsTest As Recordset
Set dbTest = openDatabase ("biblio.mdb")
Set rsTest = dbTest.OpenRecordset ("authors")
Do Until rsTest.EOF
	List1.AddItem rsTest("Author")
	rsTest.MoveNext
Loop
```

לפני ביצוע הלולאה בפעם הראשונה, בודקת התוכנית את ערך rsTest.Eof. אם התוצאה היא True (כלומר סוף ה-Recordset), הלולאה לא תבוצע כלל. זהו היתרון הנובע משילוב התנאי בתחילת הלולאה ולא בסופה.



טיפ:
שימוש בדירוג שורות הקוד מקל על זיהוי מבנים שונים בתוכנית, כדוגמת לולאות או בלוקים של פקודות If\Then\Else או Select Case\End Select. עשה זאת על ידי הקשת טאב בתחילת השורה. הקשת Enter בסוף השורה, תעביר את הסמן אל מתחת לנקודה בה התחילה השורה הקודמת (כך שאין צורך להקיש טאב בכל פעם מחדש). נסה לכתוב את שורות ההתחלה וסיום הבלוק באותה עמודה, כפי שנעשה בדוגמאות רבות בספר זה.

בדומה ללולאת Do While, גם לולאת Do Until יכולה להיכתב בצורה המאפשרת לה להתבצע לפחות פעם אחת. עיין בדוגמה הבאה:

```
Dim I As Integer
I = 0

Do
	I = InputBox("Enter 0 to exit the loop!")
Loop Until I = 0
```

למרות שהערך המוגדר מלכתחילה עבור המשתנה I הוא 0, הלולאה מבוצעת לפני בדיקת התנאי. הפעלת הקוד לעיל תגרום להופעה חוזרת של תיבת ההודעה עד אשר המשתמש יקליד 0.

לולאות מספור

סוג נוסף של לולאות, בהן ניתן להשתמש ב- Visual Basic, הן לולאות For Each. לולאות אלו מכונות **לולאות מספור** (Enumeration Loop) מכיון שהן ממספרות כל חבר במערכת אובייקטים. כפי שתלמד מאוחר יותר בפרק 16 **מחלקות: שימוש חוזר ברכיבים**, ניתן לאחסן קבוצת אובייקטים במבנה לוגי המכונה **אוסף** (Collection).

ראה: יצירת מחלקות המכילות אוספים, בפרק 16.

דוגמה אחת היא האוסף Printers. זהו אוסף מובנה המייצג, כאובייקט, כל אחת מן המדפסות המקושרות למערכת. הדוגמה הבאה מציגה כיצד ניתן להשתמש בלולאת For Each להדפסת מאפיין DeviceName של כל המדפסות:

```
Dim objPrinter As Printer
For Each objPrinter In Printers
      Debug.Print objPrinter.DeviceName
Next objPrinter
```

משתנה האובייקט ObjPrinter מתנהג בדומה למונה בלולאת For...Next, בכך שהוא מכיל בכל פעם את הפריט המעובד הנוכחי.

כמו כן, ניתן להשתמש בלולאת For Each בעבודה עם מערכים. ראה בדוגמה הבאה:

```
Dim nNumber As Variant
      Dim MyArray As Variant
      MyArray = Array(3,6,9,9,5,2,3,9)
      For Each nNumber In MyArray
            If nNumber > 5 Then Debug.Print nNumber
Next nNumber
```

תחילה יוצר הקוד מערך מסוג Variant של שמונה משתנים, ואחר כך נעזר בלולאה לרישום כל המשתנים הגדולים מ-5.

ניפוי שגיאות קוד

במהלך כתיבת הקוד והוספת אובייקטים, עליך לוודא כי לא נפלו שגיאות בפרויקט Visual Basic שלך. לעיתים קרובות קורה שתוכנית, הנראית תקינה לחלוטין על פני השטח, מסרבת לפעול בכל צורה שהיא. הסיבה לכך היא שגיאה כלשהי בקוד. נניח שברצונך להשתמש בלולאת While לצורך הדפסת כל המספרים בין 1 ל-100 ב- Immediate Window.

החל פרויקט חדש מסוג Standard EXE וכתוב את שורות הקוד הבאות בשגרת האירוע Form_Load:

```
Dim I As Integer
I = 1
While I <= 100
      Debug.Print "Count=" & I
Wend
End
```

הפעל את התוכנית ותגלה מייד כי אירעה שגיאה, הגורמת לתוכנית להדפיס את הערך 1 פעם אחר פעם. למעשה, אם תניח לתוכנית לפעול ללא הפרעה, הלולאה עצמה לא תפסיק לעולם.

בעת עבודה ב-IDE, תוכל ללחוץ על לחצן Stop, לתקן את הקוד הפגום ולהפעיל את התוכנית מחדש. אולם, יש מקרים, כמו המקרה הנוכחי, בהם תרצה לנפות שגיאות מתוכניתך במהלך פעולתה. לשם כך עליך להשתמש בלחצן Break (Pause), הממוקם בסרגל Debug, או לחילופין, להקיש Ctrl + Break. אם תוכניתך עדיין פועלת, הקש כעת Ctrl + Break. שים לב כי הופסק ביצוע הקוד (שורות חדשות אינן מודפסות יותר בחלון Immediate), אך התוכנית עוד לא עצרה לגמרי. הטפסים עדיין טעונים לזיכרון והמשתנים מחזיקים בערכיהם. במצב זה תוכל להקיש F5 או ללחוץ על לחצן Start כדי להמשיך את פעולת התוכנית. אם תציג במצב זה את חלון הקוד, תמצא כי Visual Basic מאירה את הפקודה הבאה המיועדת לביצוע ומציבה חץ קטן משמאלו.

מצב זה של עצירת ביניים מכונה **מצב שבירה** (Break Mode). כאשר אתה עובד במצב שבירה, באפשרותך לצפות או לשנות ערכי משתנים בחלון Immediate. במקרים מסוימים תוכל אף לשנות את הקוד ולהמשיך מייד בביצוע התוכנית.

טיפ:

הדרך הקלה ביותר לדעת אם הנך עובד במצב שבירה היא להציץ בכותרת של Visual Basic ולבדוק האם מופיעה בה המילה [break]. כמו כן תוכל להביט במצב הלחצנים בסרגל הכלים או לנסות לכתוב בחלון Immediate.

הערה:

מומלץ לתקן שגיאות כאלו לפני הידור התוכנית. משום שהמשתמשים מפעילים את התוכנית מחוץ לסביבת Visual Basic, וביכולתם לשבור או להפסיק לולאה אינסופית רק באמצעות סיום פעולת התוכנית כולה, מתוך חלון Task Manager של Windows.

בדוגמה זו "שכחת" להוסיף פקודה אשר תגדיל את ערך המשתנה I. כיון ש-I נשאר ללא שינוי, התנאי שהצבנו בפקודה While לא יקבל לעולם ערך False והלולאה תתבצע אינסוף פעמים. כדי לתקן את השגיאה, כאשר הינך נמצא במצב שבירה, הוסף את השורה החסרה. הקוד שלך אמור עתה להיראות כך:

```
Dim I As Integer
I = 1
While I <= 100
      Debug.Print "Count=" & I
      I = I + 1
Wend
End
```

עתה, הקש F5 או לחץ על לחצן Start של Visual Basic. התוכנית אמורה כעת למנות מ-1 עד 100 ולעצור.

הערה:

ישנן פקודות מסוימות שלא ניתן להוסיף לתוכנית במצב שבירה. במקרים אלה, Visual Basic מודיעה לך שעליך לעצור ולהפעיל את התוכנית מנקודת ההתחלה.

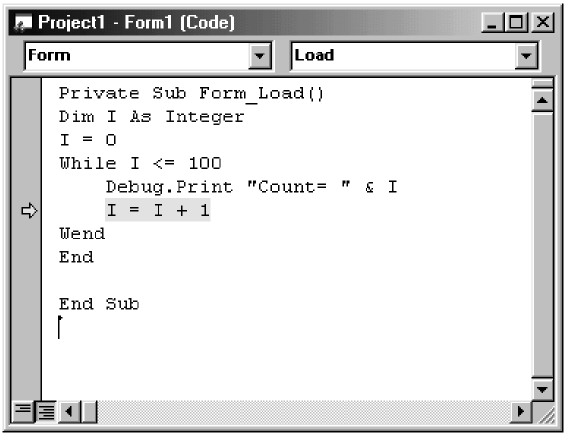
כניסה אל הקוד

במצב שבירה עומדות לפניך אפשרויות נוספות מלבד הפעלה מחדש של התוכנית. אתה יכול להיכנס אל הקוד ולעבור שורה אחר שורה, או לדלג על פני קטעים שלמים. ישנן שלוש דרכים שונות לכניסה לקוד ב- Visual Basic, כולן מתחילות בסרגל Debug:

- Step Over. מעבר על פני השורות בשיגרה הנוכחית בלבד, מבלי לעבור לשגרות אחרות הנקראות על ידה (מקשי קיצור: Shift + F8).
- Step Into. מעבר על פני השורות בשיגרה הנוכחית כולל מעבר אל השגרות הנקראות על ידה (מקש קיצור: F8).
- Step Out. הפעלת הקוד עד סוף השורה האחרונה בשיגרה הנוכחית (מקשי קיצור: Ctrl + Shift + F8).

לשם הדגמה, הפעל את תוכנית הדוגמה עד לסיומה ולאחר מכן הקש F8 מספר פעמים. שים לב כי בכל פעם שאתה מקיש F8 התוכנית מבצעת שורה אחת ומאירה את הפקודה הבאה בחלון הקוד, כפי שניתן לראות בתרשים 10.1.

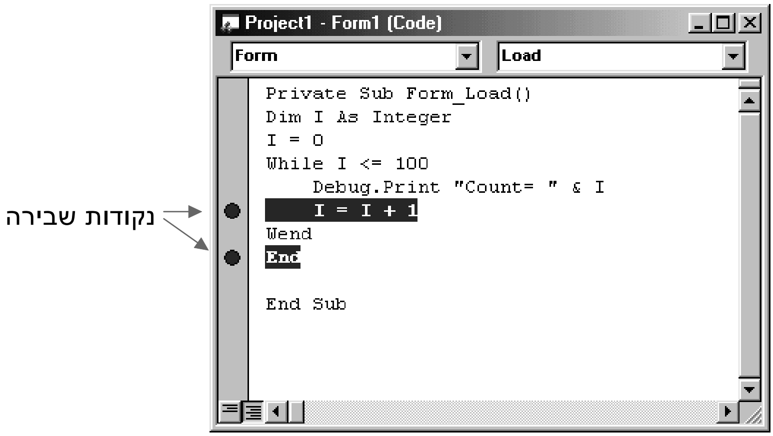
אם ברצונך לדלג על שורות קוד או לבצען מחדש לשם בדיקה, תוכל להשתמש בפקודה Set Next. מקם את הסמן על השורה הרצויה והקש Ctrl + F9 או בחר Debug, Set Next Statement. כמו כן, תוכל לגרור את החץ המופיע על השוליים השמאליים ולמקמו סמוך לשורה שברצונך לבצע.



תרשים 10.1: במצב ניפוי שגיאות, Visual Basic מאפשרת לך לערוך את הקוד שלך על גרירת החץ על השוליים השמאליים ומיקומו לצד הפקודה הנבדקת

נסה זאת כעת: היכנס לקוד תוכנית הדוגמה ומקם את החץ סמוך לשורה המעלה את ערך המשתנה I. לאחר מכן, גרור את החץ חזרה אל פקודת Print ובצע אותו על ידי הקשה על F8. שים לב כי פעולה זו גרמה לכך שפעמיים יודפס ערך זהה עבור I.

בדוגמה לעיל, ביצעת ניפוי שגיאות בלולאת While. נניח כי אינך מעוניין לנפות שגיאות מן הלולאה עצמה אלא מהקוד שלפניה ואחריה. כניסה אל תוך הלולאה, במקרה זה, יכולה להיות מטרד רציני, בעיקר אם היא אמורה להתבצע 1000 פעמים. לכן, עליך להיעזר ב**נקודות שבירה** (BreakPoints). נקודות שבירה הן שורות קוד, המיועדות להכניס את Visual Basic למצב שבירה טרם ביצוען. נקודות שבירה מאפשרות לך לבצע את התוכנית בצורה רגילה, ובשלב מסוים ליצור מצב שבירה ולהיכנס אל הקוד בנקודה מוגדרת מראש.



תרשים 10.2: נקודות שבירה מאפשרות לך לבצע את התוכנית עד לנקודה מסוימת. בשלב זה באפשרותך לעצור, לבדוק ערך משתנה מסוים או שינויים כלשהם ולהמשיך אל נקודת השבירה הבאה

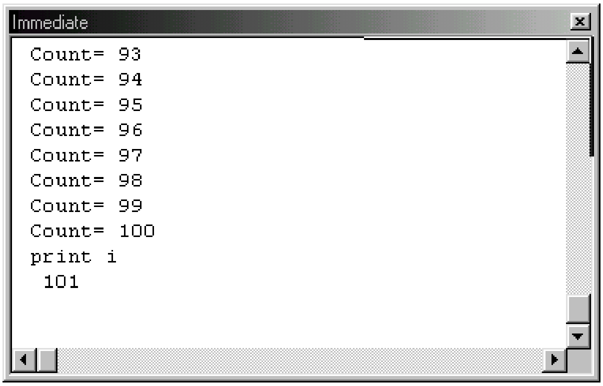
כדי ליצור נקודת שבירה, האר את שורת הקוד בה אתה מעוניין, במצב עיצוב׳ ולאחר מכן הקש F9 או בחר Debug, Toggle Breakpoint. Visual Basic מאירה את נקודת השבירה שהגדרת, ומציבה עיגול אדום משמאלה, כפי שניתן לראות בתרשים 10.2.

עתה, נסה ליצור נקודת שבירה בקוד שיצרת. הגדר את פקודת End כנקודת שבירה. סמן את הפקודה והקש F9. אחר כך, לחץ על לחצן Start (או הקש F5), כאילו ברצונך לבצע את התוכנית באופן הרגיל. כעת אמורה התוכנית לבצע את לולאת While ומייד אחר כך להיכנס למצב שבירה. הדפס את ערך המשתנה I (התוצאה האמורה להתקבל היא 101), כדי לוודא שלולאת While אכן בוצעה כמתוכנן. כעת, הקש F5 כדי להמשיך בביצוע התוכנית.

עבודה בחלון Immediate

חלון Immediate (Immediate Window), המוכר גם כחלון ניפוי השגיאות, מאפשר לך להכניס משפטים לתוך הקוד שלך, בשעה שהתוכנית נמצאת במצב שבירה. כדי להציג את החלון, ודא כי התוכנית במצב שבירה והקש Ctrl + G. כדי להמשיך בביצוע התוכנית, הקש F5 או עבור לשורת הקוד הבאה.

תרשים 10.3 מציג מספר שימושים נפוצים בחלון Immediate.

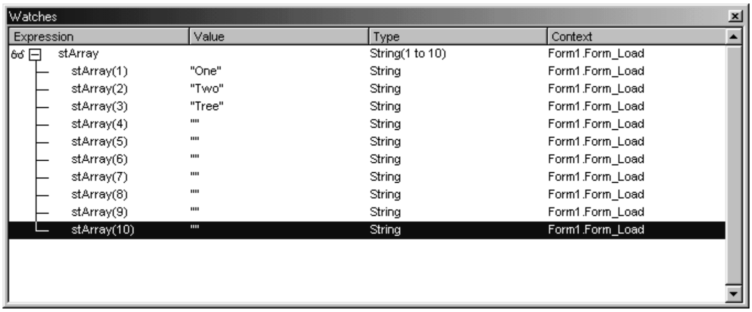


תרשים 10.3: ניתן להציב ערך במשתנה **ולקרוא לתוכנית באופן ידני**

תרשים 10.3 מדגים כיצד ניתן להציב ערך במשתנה ולקרוא לתוכנית באופן ידני. כמו כן, אתה יכול לסיים את התוכנית באמצע ביצועה על ידי הקלדת הפקודה End בחלון Immediate.

מעקב אחר ערכי משתנים

בעת עבודה במצב שבירה, מקובל כפי שזה עתה למדת, להדפיס ערכי משתנים בחלון Immediate. עם זאת Visual Basic מצוידת בשתי תכונות נוספות המאפשרות לך לעקוב אחר ערכי המשתנים בתוכנית במהלך ביצועה. הראשונה היא תכונה פשוטה: כל שעליך לעשות הוא למקם את מצביע העכבר על שם המשתנה. פעולה זו תגרום להופעת תיאור כלי (ToolTip) המציג את ערך המשתנה. התכונה השנייה, מתקדמת יותר, עושה שימוש ב**חלון Watches** (Watches Window).



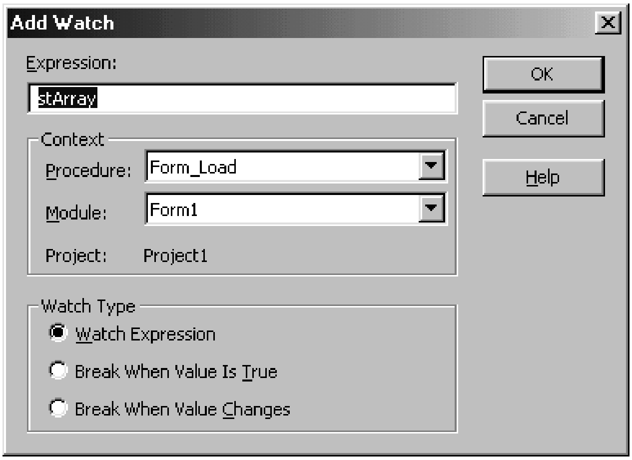
תרשים 10.4: באפשרותך לצפות ולשנות ערכי משתנים בחלון Watches

אם ברצונך לעקוב אחר ערך משתנה בחלון Watches, לחץ לחיצה ימנית על שם המשתנה ובחר Add Watch מהתפריט המקוצר. נסה זאת כעת: התחל פרויקט חדש מסוג Standard EXE והקלד את הקוד הבא בשגרת Form_Load של הטופס:

```
Private Sub Form_Load()
	Dim stArray(1 to 10) As String

	stArray(1) = "One"
	stArray(2) = "Two"
	stArray(3) = "Three"
End Sub
```

במצב עיצוב עדיין, לחץ לחיצה ימנית על **stArray** ובחר **Add Watch**. פעולה זו תגרום להופעת תיבת הדו-שיח **Add Watch** המוצגת בתרשים 10.5. לחץ **OK** והוסף יציאה עבור stArray בחלון Watches. שים לב כי בעמודת Value מופיע הביטוי <Out Of Context>, זאת משום שהתוכנית עדיין לא הופעלה והמערך עוד לא קיים. הקש **F8** פעם אחת כדי להפעיל את התוכנית וודא כי חלון Watches מופיע על המסך. אם החלון אינו מופיע, בחר **View**, **Watch Window**. הקש **F8** מספר פעמים, ובדוק בכל פעם את חלון Watches. ערכי כל האלמנטים במערך צריכים כעת להופיע בחלון. תוכל ללחוץ לחיצה כפולה על ערך המשתנה ולהכניס ערך חדש (פעולה זו זהה להצבת ערכים למשתנים בחלון Immediate). שים לב כי משתנים מרובי מימדים (כגון מערכים או אובייקטים), מופיעים בחלון במבנה היררכי של עץ. התכונה שימושית בעבודה עם עצמים מורכבים, כגון מערך רשומות (Recordset).



תרשים 10.5: תיבת הדו-שיח Add Watch מאפשרת להוסיף משתנים לחלון Watches

איתור שגיאות

במהלך כתיבת הקוד Visual Basic מסבה את תשומת לבך לשגיאות תחביריות. אולם, בעת ביצוע התוכנית עלולות, מסיבות רבות, לצוץ אל פני השטח שגיאות בלתי צפויות. לדוגמה, נניח כי ברצונך לפתוח קובץ טקסט אשר נמחק על ידי המשתמש. כאשר מתרחשת שגיאה מסוג זה בתוכנית מהודרת, מופיעה על המסך הודעת שגיאה, כפי שמראה תרשים 10.6, והתוכנית נעצרת.



תרשים 10.6: באפשרותך למנוע שגיאות בלתי צפויות ועצירות "לא סימפטיות" על ידי הוספת שגרות לטיפול בשגיאות (Error Handling) לתוכניתך

אומנם אין באפשרותך לצפות מראש את כל התסריטים היכולים להוביל להתרחשות שגיאות בתוכנית. עם זאת, קל למדי לטפל ולמנוע שגיאות מסוג File Not Found, כמו זו המופיעה בתרשים 10.6. אם לא כתבת קוד מיוחד לפתרון הבעיה, תוכל לפחות להציג הודעה "המשכילה" את המשתמש בטרם תסיים התוכנית את פעולתה.

שימוש בפקודה On Error

השיטה הנפוצה ביותר לטפל במצבי שגיאה היא שימוש בפקודה **On Error** של Visual Basic. כאשר מתרחשת שגיאה, הפקודה On Error מפסיקה את שטף התוכנית ומפעילה קוד המיועד לטיפול באותה שגיאה. להלן דוגמה טיפוסית:

```
On Error Goto FileOpenError
```

כעת, אם תתרחש שגיאה באחד המשפטים הבאים אחרי פקודת On Error, יופסק ביצועו השוטף של הקוד והתוכנית תדלג למשפט FileOpenError.

הוספת תוויות שורות קוד

הנושא בו אנו עוסקים מוביל אותנו לכלי תכנותי נוסף: **הוספת תוויות שורה** (Line Labeling). אם עבדת בעבר על גרסאות ישנות של Basic, בוודאי מוכר לך השימוש במספרי שורות, כגון:

```
10 Print "Hello"
20 Goto 10
```

תוויות השורה של Visual Basic דומות למספרי השורות הזכורים מ-Basic. תווית יכולה להכיל גם טקסט, בנוסף על ספרות, אך עליה להיות ייחודית בתוכנית. לכל תווית מצורף הסימן (:), כמו בדוגמה הבאה:

```
Privete Sub Form_Load()
        On Error Goto FileOpenError
                open "C:\NOFILE.TXT" For Input As #1
                Line Input #1, sData
                Exit Sub
FileOpenError:
        MsgBox "There was a problem opening the file. Call the Help Desk!"
        End
End Sub
```

אם אחת הפקודות: Open או Line Input תגרום להתרחשות שגיאה, התוכנית תבצע את המשפטים שלאחר התווית FileOpenError. במקרה זה תוצג על המסך הודעת השגיאה והתוכנית תסיים את פעולתה.

בדוגמה לעיל, יש מספר נקודות ראויות לציון: ראשית, שים לב למיקומה וסגנונה של שגרת הטיפול בשגיאות. נהוג למקם אותה בסמוך לסוף השיגרה אליה היא מסונפת. בשונה משאר שגרות המשנה, שם השיגרה אינו מורחק מן השוליים כדי להדגיש את ייחודה. שנית, וחשוב יותר, שים לב למיקום פקודת Exit Sub לאחר פקודת Open. חשוב למנוע את ביצוע שגרת הטיפול בשגיאות אם פקודת Open בוצעה בהצלחה.

שליטה בשטף התוכנית לאחר שגיאה

בדוגמה הקודמת סיימה התוכנית את פעולתה בעקבות התרחשות שגיאה. עם זאת, עומדות לרשותך גם דרכים אחרות (טובות יותר) לטיפול בשגיאות:

- צא משגרת המשנה, לאחר שהודעת למשתמש כי התרחשה שגיאה, והמשך בביצוע התוכנית במגבלות הקיימות.
- חדש את ביצוע התוכנית מהפקודה הבאה אחרי הפקודה שגרמה לשגיאה.
- אפשר למשתמש לתקן את השגיאה, ונסה לבצע את הפקודה הבעייתית מחדש.

תוכל לשלב כמה תוויות בתהליך אחד ולשנות שגרת שגיאות מסוימת מספר פעמים. לדוגמה, תוכל להוסיף אחרי פקודת Open תווית עבור שגרת שגיאות נוספת בשם FileInputError. תוכל גם לנטרל את תהליך הטיפול בשגיאות באמצעות הפקודה:

```
On Error Goto 0
```

פקודת On Error הולכת יד ביד עם פקודת Resume. הדוגמה הבאה גורמת לתוכנית להתעלם משגיאות ולהמשיך בביצוע שורה אחר שורה:

```
On Error Resume Next
```

עליך להשתמש בפקודה זו בזהירות רבה, הואיל והיא גורמת לתוכנית להתעלם משגיאות במקום להתמודד איתן. מומלץ יותר להשתמש ב-Resume כדי לגרום לתוכנית לדלג לחלק אחר בקוד, כפי שמראה הדוגמה הבאה:

```
Private Sub Form_Load()
    On Error Goto FileOpenError
RetryHere:
    Open "C:\NOFILE.TXT" For Input As #1
    Line Input #1, sData
    Exit Sub
FileOpenError:
    Dim sMessage As String
    sMessage = "There was a problem opening the file. "& vbCrLf
    sMessage = sMessage & "Press Retry to try again, or Cancel to quit".
    If msgBox (sMessage, vbRetryCancel + vbCritical, "Error"!) = vbRetry Then
        Resume RetryHere
    Else
        End
    End If
End Sub
```

הדוגמה לעיל מציגה תיבת הודעה המכילה לחצני Retry ו-Cancel. אם המשתמש בוחר Retry התוכנית מבצעת את הקוד המסונף לתווית RetryHere ומבצעת מחדש את פקודת Open.

קביעת סוג השגיאה

בעת התרחשות שגיאה, הקוד שלך יכול לברר פרטים נוספים לגבי סוג השגיאה ונסיבותיה באחת הדרכים הבאות:

- Err - מכיל מספר המייצג את השגיאה.
- Error - מכיל מחרוזת לתיאור השגיאה.
- Err Object - מכיל את מספר השגיאה, תיאור ומידע נוסף. משמש גם להגדרת שגיאות מותאמות אישית. כמו כן יש לו מאפיינים שמייצגים את מספר השגיאה: Err.Number ואת תיאורה: Err.Description.

אם אתה יודע כיצד לתת פתרון לשגיאות מסוימות, תוכל להשתמש באובייקטים אלה כמענה נבון. בדוגמה האחרונה אירעה שגיאת File Not Find המיוצגת על ידי המספר 53. תוכל בקלות להוסיף לשגרת השגיאה קוד אשר ייתן מענה מתאים (כלומר, בדיקת קובץ אחר) אם הערך של Err הוא 53.

טיפ:
כאשר אתה כותב שגרת שגיאות העושה שימוש בתיבת הודעה, הצג בתיבת ההודעה את מספר השגיאה ותיאור קצר שלה, זאת כדי להקל על תהליך הטיפול בשגיאות.

מכאן...

פרק זה ערך עבורך סקירה כללית בנושא תכנות ב- Visual Basic. למדת כיצד לטפל במחרוזות, בקבלת החלטות ובלולאות בתוכניתך. חומר נוסף בנושא תכנות תוכל למצוא בפרקים הבאים:

- מידע נוסף על Visual Basic תמצא בפרק 9 **יסודות התכנות של Visual Basic**.
- מידע בנושא יצירת פונקציות תוכנית משלך תמצא בפרק 11 **ניהול הפרויקט: תת-שגרות, פונקציות וריבוי טפסים**.
- מידע שיעזור לך לשפר את יכולות עיצוב הממשק שלך תמצא בפרק 18 **תכנון ממשק נכון**.

11

ניהול הפרויקט: תת-שגרות, פונקציות וריבוי טפסים

מה בפרק?

- השימוש בשגרות ובפונקציות
- עבודה עם טפסים מרובים
- ניהול רכיבים בפרויקט
- שליטה בהתחלת תוכנית

בפרק 9, ***יסודות התכנות של Visual Basic***, למדת איך לכתוב קודים שיגרמו לתוכנית לבצע מטלות שונות. הודגם שם כיצד לתפעל נתונים, ואיך משפטי בקרה מאפשרים ביצוע פעולות החוזרות על עצמן והפעלת משפטי קוד באופן סלקטיבי. אולם, יצירת תוכנית טובה, הניתנת לטיפול ולתחזוקה, דורשת יותר מאשר כתיבת קודים.

אחת ממטלות המפתח היא יצירת קטעי קוד וחלקי תוכנית הניתנים לשימוש חוזר, כדי שלא יהיה צורך להמציא מחדש את הגלגל (או במקרה זה, את תוכנית המחשב). מיומנות חשובה נוספת היא הכושר לנהל ביעילות את קטעי הקוד והטפסים. פרק זה דן בשני ההיבטים הללו של ניהול פרויקט תכנות. ראשית תלמד איך ניתן להשתמש ב**שגרות**, לביטול הצורך לכתיבת קטעי קוד החוזרים על עצמם. אחר-כך נבחן כיצד מוסיפים שגרות אלו ורכיבי תוכנה אחרים לפרויקט. בסוף הפרק תינתן סקירה קצרה לגבי בניית התוכנית והפצתה לשימוש אחרים.

השימוש בשגרות ובפונקציות

לעיתים קרובות, כשמפתח כותב תוכניות רבות וגדולות יותר, הוא מוצא את עצמו משתמש באותו קטע קוד בכמה מקומות באותה תוכנית או חבילת תוכניות. האם קיימת דרך יעילה יותר לשימוש חוזר בקוד באותה תוכנית, מלבד הצבתו החוזרת פעמים אחדות באותה תוכנית? בוודאי שיש. הפתרון הוא בשימוש ב**שגרות** (Procedures) וב**פונקציות** (Functions) (שגרות מתחלקות לשגרות אירוע הנוצרות על ידי המחשב ומגיבות לאירועים ידועים מראש, ושגרות משתמש הנוצרות על ידי המשתמש). שגרות ופונקציות הם מקטעי תוכנית המבצעים מטלה מסוימת, ואחר כך מחזירים את העיבוד לאותו אזור בקוד ממנו נקראו (פונקציות מחזירות ערכים ואילו שגרות לא). פרוש הדבר שניתן לקרוא לשיגרה (או פונקציה) מסוימת ממספר מקומות בקוד, וכן ניתן בניהול נכון, להשתמש בשיגרה מסוימת ביותר מתוכנית אחת.

הקורא כבר נחשף לעבודה עם שגרות, גם אם לא ידע זאת. כל הזנת קוד כתגובה לאירוע, בהפעלה על ידי לחצן פקודה (או פקד אחר), מהווה למעשה בניית **שגרת אירוע** (Event Procedure). שגרות אירוע נקראות על ידי התוכנית בכל פעם שקורה אירוע. כידוע, יכול המשתמש לבנות בעצמו שגרות ולקרוא להן לפי הצורך. שגרות אלו נקראות **תת-שגרות** (Subroutines) המוגדרות על ידי המשתמש. למרות שהקודים בדוגמאות הקודמות היו כולם כלולים בשגרות אירוע, תוכנית "אמיתית" יכולה לכלול קוד נוסף בתת-שגרות המוגדרות על ידי המשתמש.

עבודה עם שגרות

רעיון המפתח מאחורי העבודה עם שגרות הוא פירוק התוכנית לסדרת מטלות קטנות יותר. כל אחת ממטלות אלו ניתנת ל״אריזה״ בשיגרה, פונקציה, או אולי מחלקה (מחלקות נדונות בפרק 16 **מחלקות: שימוש חוזר ברכיבים**). לתכנות בדרך זו יש יתרונות אחדים:

- ניתן לנסות כל מטלה לחוד. כמות הקוד הקטנה יותר בשיגרה מפשטת את תהליך איתור התקלות, ומקלה על עבודת מספר מפתחים על אותו פרויקט.
- ניתן למנוע כתיבת קטעי קוד החוזרים על עצמם, על ידי קריאה לשיגרה בכל פעם שיש לבצע מטלה מסוימת, במקום לחזור על כתיבת הקוד.
- מפתח יכול לבנות ספריית שגרות שניתן להשתמש בה ביותר מתוכנית אחת, ובכך לחסוך זמן פיתוח בפרויקטים חדשים.
- תחזוקת התוכנה פשוטה יותר מכמה סיבות: אם אין חוזרים על הקוד, יש לערוך ולהגיה אותו רק פעם אחת. בנוסף, ההפרדה בין רכיבי המפתח (למשל ממשק המשתמש ופונקציות בסיס הנתונים) מאפשרת לבצע שינויים גדולים בתוכנית, מבלי לשנות את כל הקודים.

טיפ:

כדי להתאים את הקוד לשימוש חוזר, כדאי להשתמש בהערות לעיתים תכופות. הרחבה זו בפרטים תאפשר למפתח אחר (או למפתח עצמו לאחר תקופת זמן ארוכה), להבין במהירות מה מטרת כל שיגרה ואיך היא מושגת.

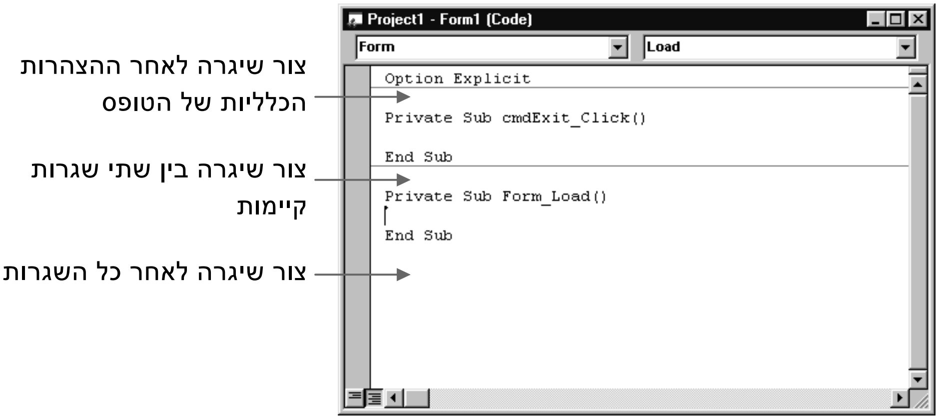
יצירת שיגרה

כמו במקרה של תוכנית שלמה, תהליך יצירת השיגרה מתחיל בתכנון. יש צורך לקבוע מה המשימה האמורה להתבצע, איזה מידע יידרש על ידי השיגרה, ואיזה מידע יוחזר על ידה. לאחר השלמת משימה זו, ניתן להתחיל בקידוד השיגרה. בשפת Visual Basic ניתן להתחיל בבניית שיגרה באחת משתי דרכים: להתחיל מלא-כלום, או להשתמש בתיבת הדו-שיח **Add Procedure**. שתי השיטות קלות יחסית, והבחירה היא עניין של העדפה אישית. כמו כן, בשתי השיטות חייב המשתמש להימצא בחלון Code לפני שיוכל לבנות את השיגרה.

ליצירת שיגרה מהיסוד, הצב את הסמן בחלון Code, במיקום שאינו בתחום פונקציה או שיגרה שכבר הוגדרה. ניתן למשל להציב את הסמן לאחר השורה End Sub בשיגרה, לפני השורה Sub של שיגרה אחרת, או בראש החלון Code. תרשים 11.1 מראה היכן ניתן להתחיל.

טיפ:

הצבת השגרות בסדר אלף-בית מקלה על איתורן תוך גלילה בחלון ה-Code. הן תמיד תופענה בסדר האלף-בית בתיבה הנפתחת **Procedure** שבחלון Code.

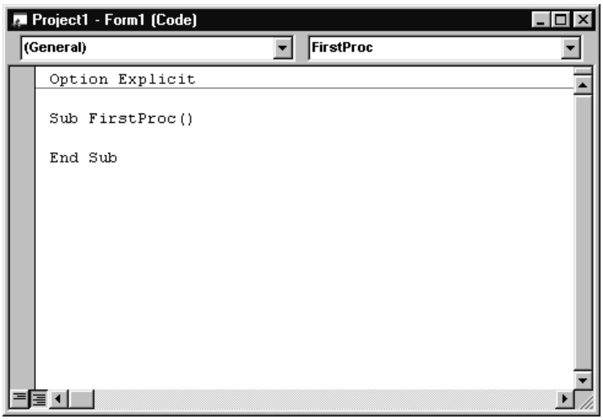


תרשים 11.1: יצירת שיגרה בחלון Code בשורת Sub

יצירת שיגרה חדשה בחלון Code מתבצעת על פי הצעדים הבאים:

1. פתח את הקוד עבור Form1 על ידי לחיצה על הלחצן **View Code**.
2. הקלד את מילת המפתח **Sub**, ולאחריה רווח.
3. הקלד את שם השיגרה החדשה, בדוגמה זו FirstProc.
4. הקש Enter ליצירת השיגרה.

בהקשת Enter קורים שלושה דברים: זוג סוגריים נוסף בקצה שורת Sub, שורת End Sub מוצבת בחלון Code, והאובייקט הנוכחי ברשימה הנפתחת Procedure של חלון עריכת הקוד, הופך להיות שם השיגרה. תרשים 11.2 מראה שינויים אלה בשיגרה הנקראת FirstProc.



תרשים 11.2: השורה End Sub נוספת אוטומטית עם הגדרת שיגרה חדשה

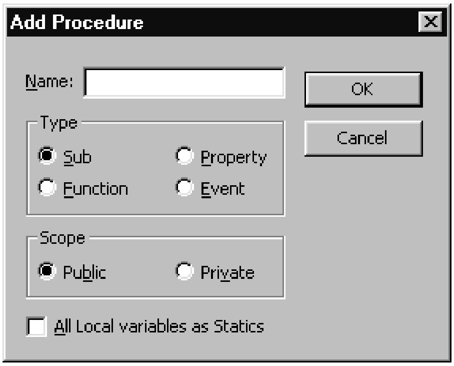
עכשיו אתה מוכן להזנת כל פקודה שברצונך להריץ כשהשיגרה נקראת.

תחביר שגרת Sub כולל את השורה Sub, השורה End Sub, ואת פקודות השיגרה :

```
[Friend | Public | Private] [Static] Sub procname([arguments])
statements_to_be_run
End Sub
```

מילות המפתח Public, Private, Friend ו-Static בשורת Sub הן רשות, ומשפיעות על המיקומים מהן יכולה השיגרה להיקרא. מילות מפתח אלו מצביעות על **טווח ההכרה** (Scope) של השיגרה באותה הדרך שבה הן מצביעות על טווח ההכרה של המשתנה.

השיטה האחרת ליצירת שיגרה הוא שימוש בתיבת הדו-שיח **Add Procedure** (ראה תרשים 11.3). הגישה לתיבת דו-שיח זו היא בבחירת Tools, Add Procedure.



תרשים 11.3: ניתן גם ליצור שיגרה חדשה בשימוש בתיבת הדו-שיח Add Procedure

ליצירת מעטפת השיגרה, בצע בתיבת הדו-שיח Add Procedure את הצעדים הבאים :

1. הכנס את שם השיגרה בתיבת הטקסט **Name**.
2. בחר את סוג השיגרה (Sub, Function, Property, Event).
3. בחר את טווח ההכרה של השיגרה (Public, Private).
4. אם דרוש, סמן את תיבת הסימון All Local Variables as Statics.

ליצירת שיגרה יש לבחור סוג תת-שיגרה. לבחירתך ארבע אפשרויות : **פונקציה** (Function) המחזירה ערך מוגדר. שגרות אלו מכוסות בהמשך בפרק. **שגרת מאפיין** (Property) קובעת או שולפת את ערך המאפיין בצורת מודול מחלקה. **שגרת אירוע** (Event) מגיבה לאירוע בטופס או במודול מחלקה. בחירת אירוע בתיבת דו-שיח זו מוסיפה הצהרה מקדימה לאירוע מוגדר-משתמש. השגרות Property ו-Event מתוארות בפרק 16, מחלקות : שימוש חוזר ברכיבים.

לאחר הזנת הנתונים הדרושים, לחץ OK. Visual Basic תיצור את מסגרת השיגרה.

הרצת שיגרה

לאחר פיתוח השיגרה, דרושה דרך להריץ אותה מתוך חלקי תוכנית אחרים. להרצת שיגרה ניתן לבחור באחת משתי שיטות: שימוש בשורת Call, או בפשטות על ידי שימוש בשם השיגרה וה**ארגומנטים** (Arguments) הנדרשים על ידה (הארגומנטים הם אלה שצוינו בשורה Sub כשהשיגרה הוגדרה).

התחביר להרצת שיגרה הוא:

```
Call procname([arguments])
```

או:

```
procname arguments
```

בשתי השורות *procname* מתייחס לשם השיגרה. שם זה מצוין בשורת Sub שהגדירה את השיגרה. *Arguments* מתייחס לפרמטרים המועברים אל השיגרה כאשר השיגרה דורשת זאת. בשורה הקוראת, יכולים הארגומנטים להיות ערכים מילוליים, משתנים, או פונקציות המחזירות את טיפוס הנתונים המתאים. זאת בשונה משורת Sub, שבה הארגומנטים חייבים להיות שמות משתנים. **פרמטרים** (Parameters), אם הם קיימים, חייבים להיות מופרדים בפסיקים.

נבחן עכשיו דוגמה קצרה לשיגרה המשתמשת בפרמטרים. נניח שהתוכנית רוצה להציג על המסך טקסט כלשהו.

```
Sub Message(sMessage As String)
    MsgBox sMessage
End Sub
```

שורת הקוד הבאה קוראת לשיגרה. כשקוראים לשיגרה ניתן לספק ערכים לארגומנטים שלה תוך שימוש במשתנה, מחרוזת מילולית, או שילוב של השניים:

```
Message "This is my message"
```

שגרת Message פשוטה, ויחד עם זאת חוסכת זמן רב בטווח הארוך. היא הופכת את הקוד הקורא לה לקצר וקריא יותר.



אזהרה:

באופן אופייני חייב מספר הפרמטרים בשורה הקוראת לשיגרה להיות שווה לאלה הקיימים בהגדרתה. כמו כן, הערכים המסופקים על ידי השורה הקוראת חייבים להתאים לסוגי הנתונים שהשיגרה מצפה להם. הפרת כל אחד מכללים אלה תגרום לשגיאה בעת הרצת התוכנית.

בתחילת קטע זה הזכרנו שתי שיטות הבאות בחשבון לקריאת שיגרה. שורת הקוד הבאה קוראת לשיגרה Message, בשימוש בתחביר השיטה האחרת:

```
Call Message("The server was rebooted")
```

כפי שניתן לראות, השימוש במילת המפתח Call אינו חובה בעת הקריאה לשיגרה. אם משתמשים בפקודה Call, חייבים לכלול את הפרמטרים בתוך סוגריים. מומלץ להשתמש בתחביר שאיננו משתמש ב-Call. גישה זו מקלה על ההבחנה בין קריאת שיגרה וקריאת פונקציה, כפי שניתן לראות כשבוחנים את הדוגמאות בסעיף הבא.

העברת נתונים לשיגרה

בשתי דרכים ניתן להעביר לשיגרה נתונים לצורך עיבוד: אפשר להגדיר את המשתנים כמשתנים **ציבוריים** (Public), הזמינים בכל מקום בתוכנית, או ניתן להעביר את המשתנים ישירות לשגרות במשפטי הקריאה.

לדוגמה, ניתן להוסיף ארגומנט שני לשיגרה Message, שיאפשר לה להציג לחצנים שונים:

```
Sub Message(sMessage As String, Button As Integer)
      MsgBox sMessage, Button
End Sub
```

אולם, משמעות גישה זו היא, שגם אם נבחר תמיד באותו אופן תצוגה של לחצנים נצטרך תמיד להעביר את המשתנה Button:

```
Message "My Message ", 1
```

בתוכנית מסוימת זו הארגומנט Button כנראה איננו משתנה במשך ההרצה. אולם, קידוד קשיח שלה לתוך השיגרה Message גם הוא איננו סביר, כך ששימוש במשתנה ציבורי נראה כבחירה הגיונית:

```
Public Button As Integer

Sub Message(sMessage As String)
      MsgBox sMessage, Button
End Sub
```

התוכנית צריכה, לפני שהיא קוראת לשיגרה, לקבוע את ערכו של Button. מילת המפתח Public גורמת לכך שצורך זה הוא יהיה גלוי לכל השגרות האחרות בתוכנית.

אם מתכוונים להשתמש במשתנים בכמה שגרות, והשיגרה הנקראת ספציפית לתוכנית הנוכחית, קביעת המשתנים כמשתנים ציבוריים היא הגישה היותר טובה. אולם, לשם הבטחת שימושיותה של שיגרה בפרויקטים רבים, הקפדה ככל האפשר על עצמאות השגרות היא רעיון טוב. לשם כך יש להגדיר את כל הפרמטרים שיש להעביר לשיגרה בשורה Sub, ולהעבירם בשורה הקוראת. בנוסף, כל המשתנים בהם משתמשת השיגרה צריכים להיות משתנים **מקומיים** (Private), המוכרזים מתוך השיגרה עצמה.

הפרמטרים המשמשים בשיגרה יכולים לספק תקשורת דו-כיוונית בין השיגרה לתוכנית הקוראת. השיגרה יכולה להשתמש במידע הכלול בפרמטרים לביצוע החישובים, ואז להחזירם לתוכנית הקוראת בפרמטר אחר.

למשל, שיגרה הבאה מקבלת את גובהו של מלבן ואת רוחבו מרשימת הפרמטרים, ומחשבת את שטח המלבן ואת היקפו. ערכים אלה מוחזרים דרך רשימת הפרמטרים:

```
Sub CalcRectangle(nWidth as Integer, nHeight as Integer, _
          nArea as Integer, nPerimeter as Integer)
     nArea = nWidth * nHeight
     nPerimeter = 2 * (nWidth + nHeight)
End Sub
```

ניתן לקרוא לשיגרה על ידי שימוש במשתנים לפרמטרים של קלט ופלט כדלקמן:

```
nWid = 5
nHgt = 5
nArea = 0
nPerm = 0
CalcRectangle nWid, nHgt, nArea, nPerm
```

ניתן גם לקרוא לשיגרה על ידי שימוש בערכים מילוליים לפרמטרים של הקלט ושימוש במשתנים לפרמטרים של הפלט:

```
nArea = 0
nPerm = 0
CalcRectangle 4, 10, nArea, nPerm
```

העברת פרמטרים לשיגרה בדרך זו מוכרת כ**העברה על ידי ייחוס** (Passing by Reference). שם המשתנה המועבר לשיגרה ושם המשתנה שהשיגרה משתמשת, מתייחסים שניהם לאותו מיקום בזיכרון. בהצהרת פונקציה או שיגרה, ניתן לציין במפורש פרמטרים **על ידי ייחוס**, בעזרת מילת המפתח **ByRef**, כבדוגמה הבאה:

```
Sub ChangeString(ByRef AnyString As String)
     AnyString = "After"
End sub

Dim sSampleString As String
sSampleString = "Before"
ChangeString sSampleString
```

שלוש שורות הקוד הראשונות מגדירות תת-שיגרה פשוטה, המשנה תוכן מחרוזת, והיא מועברת כפרמטר. מילת המפתח ByRef מציינת שכל שינוי המבוצע בפרמטר יתבטא במשתנה המועבר לשיגרה. במילים אחרות, שינוי ערך AnyString בתוך השיגרה גורם לערך sSampleString להשתנות בהתאם. יכולת זו מאפשרת לשיגרה לשנות את הערך המוחזר אז לקוד הקורא.

ניתן להעביר פרמטר לשיגרה **לפי ערך** (By Value). גישה זו גורמת לשיגרה להשתמש *בהעתק* של המידע שהועבר אליה, ומונעת מקוד השיגרה לשנות את הערך המשמש את התוכנית הקוראת. כשמוכרז פרמטר לשיגרה, ברירת המחדל היא העברה על ידי ייחוס. כדי לשנות זאת, יש להודיע לשפת Visual Basic במפורש להעביר את הפרמטר לפי ערך. לשם כך יש להוסיף את מילת המפתח **ByVal** לרשימת הפרמטרים, לפני כל משתנה האמור להיות מועבר לפי ערך, כמודגם בקוד הבא:

```
Sub CalcRectangle(ByVal nWidth As Integer, ByVal nHeight As Integer, _
            nArea, nPerimeter As Integer)
```

אזהרה:
כשפרמטרים מועברים על ידי ייחוס, דרוש להצהיר בצורה מפורשת על המשתנים בתוכנית הקוראת ובשיגרה, ולוודא שסוגי המשתנים זהים בשניהם.

יציאה מוקדמת מהשיגרה

כשתוכניות, ולכן גם שגרות, הופכות לגדולות ומסובכות, לפעמים אין צורך שכל פקודות השיגרה תתבצענה. אם יש צורך לצאת משיגרה לפני שכל הפקודות בוצעו, ניתן להשתמש בשורה Exit Sub או לחילופין אם זו פונקציה (Function) לצאת באמצעות Exit Function. דרך אחת לשימוש בשורה Exit Sub היא להכניס בתחילת השיגרה קטע קוד, הבודק את התאמת ערכי הפרמטרים. אם פרמטר המועבר לשיגרה הוא מהסוג הלא נכון, או שערכיו יכולים לגרום לבעיות בשיגרה, ניתן להשתמש בפקודת Exit Sub להפסיק את פעולת השיגרה לפני קרות השגיאה. שימוש מסוג זה נקרא **אימות נתונים** (Data Validation). הקוד הבא משנה את הקוד הקודם לחישוב שטח המלבן, לצורך ביצוע בדיקה זו:

```
Sub CalcRectangle(nWidth as Integer, nHeight as Integer, _
      nArea as Integer, nPerimeter as Integer)
      If nWidth <= 0 Or nHeight <= 0 Then
            Exit Sub
      End If
      nArea = nWidth * nHeight
      nPerimeter = 2 * (nWidth + nHeight)
End Sub
```

עבודה עם פונקציות

פונקציות דומות לשגרות, בהבדל מפתח אחד: הן מחזירות ערך. ערך זה יכול להיות מוצב למשתנה או לשמש בביטוי. Visual Basic מציעה פונקציות מובנות אחדות שניתן להשתמש בהן, כמו למשל הפונקציה Abs, המחזירה את הערך המוחלט של מספר, או Left, המחזירה מספר נתון של תווים מצידה השמאלי של מחרוזת. המשתמש גם יכול לבנות פונקציות בעצמו.

כמו עם שגרות, גם בבניית פונקציה עומדות לרשות המשתמש שתי אפשרויות: בניית הפונקציה מלא-כלום, או שימוש בתיבת הדו-שיח **Add Procedure**. להתחלה מן היסוד, בחר את המקום בחלון Code שבו רצונך שהפונקציה תתחיל, והכנס שם את מילת המפתח Function, כשלאחריה שם הפונקציה. המוסכמות לגבי מתן שמות לפונקציות זהות למוסכמות של שגרות. כדי להשתמש בתיבת הדו-שיח Add Procedure פשוט בחר את לחצן האפשרויות Function Type בתיבת הדו-שיח. כל אחת מהשיטות יוצרת מעטפת של פונקציה, בדיוק כפי שהיא עושה לגבי שיגרה. דוגמה ליצירת מעטפת כזאת מובאת בשורות הקוד הבאות:

```
Public Function NumAverage()

End Function
```

למרות שהשורה הראשונה היא הכרזה קבילה של פונקציה, ברוב הפעמים מגדירים את סוג הערך שיוחזר על ידי הפונקציה. מגדירים סוג זה של פונקציה כמו שמוגדרים סוגי המשתנים בשורת Dim, על ידי שימוש במילת המפתח As ולאחריה סוג המשתנה. הכרזת סוג הפונקציה באה אחרי הסוגריים הסוגרות על הכרזת הפרמטר. בנוסף, מוכרזים בדרך כלל הפרמטרים המועברים לפונקציה במשפט ההכרזה. משפט הכרזה שלם יותר מוצג בשורה הבאה:

```
Public Function NumAverage(inpt1 As Single, inpt2 As Single) As Single
```

הבדל עקרוני נוסף בין בניית פונקציה לבניית שיגרה הוא הקצאת ערך לפונקציה במקום כלשהו בתוך הקוד שלה. ערך זה חייב להיות מאותו סוג כפי שהוגדר בהצהרת הפונקציה והוא יהיה הערך שיוחזר על ידי הפונקציה למי שקרא לפונקציה. ניתן לראות זאת בשורה השנייה של הקוד הבא:

```
Public Function NumAverage(inpt1 As Single, inpt2 As Single) As Single
    NumAverage = (inpt1 + inpt2) / 2
End Function
```

הערה:

למרות שקוד פונקציה יכול להקצות ערך לפונקציה פעמים אחדות, רק הערך האחרון המוקצה לפני סיום הפונקציה (או היציאה ממנה) מוחזר על ידה.

בקריאה לפונקציה, מציבים בדרך כלל את הערך המוחזר על ידה למשתנה בתוכנית, או משתמשים בו במשפט תנאי, כמודגם כאן:

```
'Assigning a function to a variable
fAvgNum = NumAverage(25, 15)

'using a function in a conditional expression
If NumAverage(num1, num2) > 20 Then MsgBox "What an Average!"
```

הערה:

אם מה שנדרש מפונקציה זה בפשטות לבצע משימה (למשל פתיחת בסיס נתונים), ניתן לקרוא לפונקציה כפי שקוראים לשיגרה, ולהתעלם מהערך המוחזר.

קביעת טווח ההכרה של שגרות ופונקציות

כשיוצרים שיגרה (או פונקציה), רוצים לפעמים להגביל את תחומי השימוש בה ואת המשאבים המוקצים לה, כדי להפוך את הקוד לזמין לחלקים אחרים של התוכנית. התחום ממנה ניתן לקרוא לשיגרה נקרא **טווח ההכרה** (Scope) השיגרה.

ניתן להגדיר שגרות באחת משתי דרכים: כ**שיגרה ציבורית** (Public Procedure) או כ**שיגרה פרטית** (Private Procedure). מילת המפתח שבה משתמשים בשורת Sub, קובעת לאיזו תוכנית או שיגרה אחרת יש גישה לשיגרה שלך.

הערה:

טווח ההכרה בשגרות ובפונקציות קשור לטווח ההכרה במשתנים. הנושא נדון בפרק 9, **שימוש במשתנים לאחסון מידע**, בקטע **היכן ניתן להשתמש במשתנה**.

"לצאת לציבור"

אם יש כוונה שהשיגרה או הפונקציה תהיה זמינה לכל אורך התוכנית, יש להשתמש במילת המפתח Public בעת הגדרת השיגרה. Public מאפשרת לקרוא לשיגרה שהוגדרה בטופס או מודול מסוים, מכל שיגרה או טופס אחר בתוכנית. אבל יש להיזהר בבחירת שמות לשגרות ציבוריות, שכן לכל שיגרה ציבורית חייב להיות שם יחיד במינו בכל טווח ההכרה שלה.

אם מילות המפתח Public או Private אינן מופיעות בשורת Sub, ברירת המחדל היא שהשיגרה נקבעת כציבורית.

שמירת הפונקציה כפרטית

הכנסת מילת המפתח Private בשורת Sub תאפשר גישה לשיגרה רק מהטופס או המודול שבו היא הוגדרה. לגישה זו יש כמובן יתרונות וחסרונות. יתרון אחד הוא, שהשיגרה הפרטית שוכנת בזיכרון רק כשהמודול שבו היא מאוחסנת נטען אליו, תוך חיסכון במשאבי המערכת. החיסרון הוא, שאין גישה לשיגרה ממודולים אחרים.

מי שעובד עם שגרות אירוע (נדונות בפרקים אחרים בספר), יכול להבחין שכברירת מחדל, שגרות אלו תמיד פרטיות. הן פרטיות מפני שבצורה אופיינית אין גישה לפקדים מחוץ לטופס שבו הן שוכנות. זוהי דוגמה ל**הסתרת מידע** (Information Hiding) או **כמיסה** (Encapsulation), טכניקה שמשתמשים בה בתכנות מונחה-עצמים. כשמפתח משתמש במודול מסוים בשותפות עם אחרים, הוא יכול להגדיר את הפונקציות הנקראות על ידי אחרים כציבוריות, ואת אלו המשמשות רק אותו כפרטיות.

שימור משתנים

באופן אופייני כששיגרה מורצת, המשתנים שהיא משתמשת בהם או מייצרת אותם, מושמדים עם סיום הפעלתה. לפעמים אבל, יש רצון לשמור את ערכי המשתנים לקריאה עתידית על ידי השיגרה. ניתן לבצע משימה זו על ידי שימוש במילת המפתח Static. מילת מפתח זו יכולה להיות מיושמת עם הכרזת משתני השיגרה, או בשלב הכרזת השיגרה עצמה.

כאשר Static מופיעה בהכרזת משתנים, נשמרים רק המשתנים הנכללים במשפט Static. אם משתמשים במילת מפתח זו בהכרזת השיגרה, נשמרים כל משתניה. לדוגמה: בשליחת דוח למדפסת, צריכה השיגרה המייצרת את כותרת העמוד לעקוב אחרי מספרו. בדוגמת הקוד הבאה, ערך המשתנה הסטטי nPageNum נשמר בכל פעם שהשיגרה מופעלת:

```
Public Sub PrintHeader()
    Static nPageNum As Integer
    nPageNum = nPageNum + 1
    Printer.Print "FORTNER MASONRY ANNUAL EARNINGS REPORT"
    Printer.Print Date
    Printer.Print "Page " & nPageNum
    Printer.Print
End Sub
```

טיפ:

לצרכי יעילות, חשוב להציב לשיגרה את טווח ההכרה המתאים. מתן טווח רחב מדי לשיגרה (למשל הגדרת השיגרה כציבורית כשדרוש שהיא תוגדר כפרטית), מבזבזת משאבי מחשב. כשפונקציה מוגדרת כציבורית, חייבת Visual Basic להקצות לה משאבים מתאימים, כדי שאפשר יהיה להעמידה לרשות כל חלקי התוכנית. השימוש במילת המפתח Static, המחייב שיגרה "לזכור" את הערכים המקומיים, גורם גם הוא לבזבוז משאבים. בדרך כלל רצוי, אם ניתן, להגדיר שיגרה כפרטית, וכן להימנע משימוש במשתנים סטטיים. גישה כזאת נותנת ל- Visual Basic שליטה יותר יעילה בזיכרון, שכן אז היא חופשיה לשחרר קטעי קוד שונים לפי הצורך.

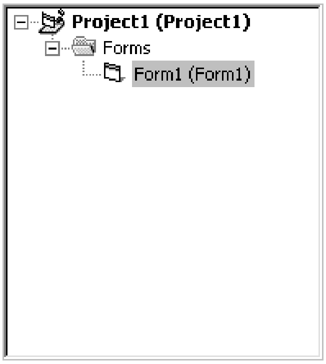
שימוש חוזר בפונקציות ובשגרות

שגרות ניתן ליצור בטופס או במודול. מיקום השיגרה תלוי במקום שבו נדרש השימוש בה, ובמטרה לה נועדה. אם השיגרה אופיינית לטופס מסוים, או אם היא משנה את מאפייני הטופס או הפקדים הקשורים בו, כדאי למקם את השיגרה בטופס עצמו.

אם, לעומת זאת, השיגרה מופעלת ביותר מטופס אחד בתוכנית, או שקיימת שיגרה כללית, המשמשת בתוכניות מרובות, יש למקם אותה במודול. המקום שבו מאוחסנת השיגרה נקבע לפי המקום שבה היא נוצרה. אם רוצים, ניתן להעביר שיגרה מטופס למודול ולהפך, תוך שימוש ב״גזור והדבק״ או אף ב״גרור ושחרר״.

אחסון פונקציה בקובץ טופס

כדי ליצור שיגרה בקובץ-טופס, יש רק לבחור את הטופס מחלון Project, ואז לגשת לקוד הטופס. ניתן לעשות זאת בלחיצה כפולה בטופס עצמו (או על כל פקד), או על ידי בחירת לחצן View Code בחלון Project. לאחר הופעת החלון Code, יוצרים את השיגרה, כמתואר לעיל בסעיף **יצירת שיגרה**.



תרשים 11.4: ניתן לבחור את הטופס לשיגרה מתוך החלון Project

שימוש בקובץ מודול עבור שיגרה

קובץ מודול מכיל רק קוד, ללא אלמנטים של טפסים או אירועים. אם קובץ מודול כבר קיים בפרויקט, ניתן ליצור שיגרה חדשה על ידי בחירת הקובץ, פתיחת חלון Code, ובניית השיגרה לפי הצעדים שתוארו לעיל.



טיפ:

לחיצה כפולה על שם המודול בחלון Project פותחת אוטומטית את חלון Code של המודול.

אם קובץ מודול איננו קיים בפרויקט, או אם רוצים להשתמש במודול חדש, ניתן ליצור מודול על ידי בחירה Project, Add Module. ניתן גם ליצור מודול חדש באמצעות לחיצה על הלחצן Add Form בסרגל הכלים, ובחירת Module מן התפריט הנפתח. בשתי הדרכים מוצגת תיבת הדו-שיח Add Module. בחר בסמל המודול, ולחץ Open. נוצר מודול חדש, ומופיע חלון Code, שניתן להתחיל בו את העריכה. עם שמירת הפרויקט או היציאה מ- Visual Basic, מתבקש המפתח לספק שם לקובץ המודול.



הערה:

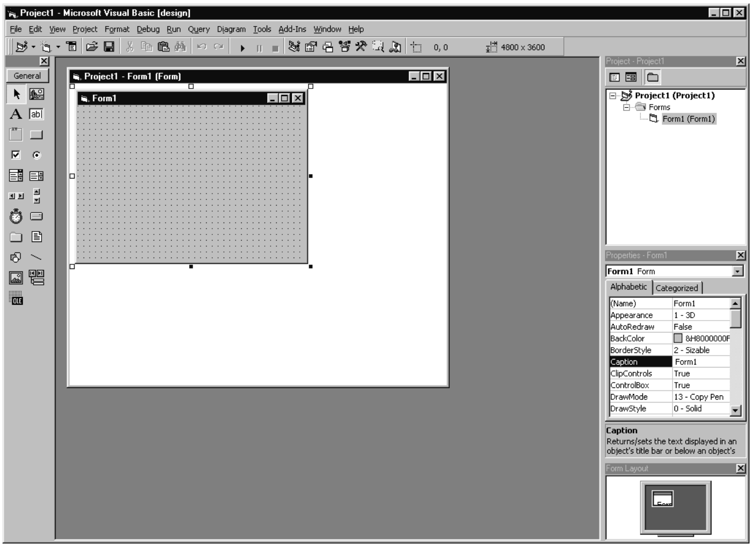
לחצן סרגל הכלים להוספת טופס או מודול חדש הוא לחצן של רשימה נפתחת. לחיצה על החץ מציגה רשימת פריטים. לאחר בחירת פריט משתנה ציור הלחצן.

עבודה עם טפסים מרובים

למרות שתוכניות פשוטות דורשות רק טופס בודד, רוב התוכניות מורכבות מטפסים מרובים. סיבה אחת לכך הוא מגבלות השטח הזמין בטופס בודד. סיבה אחרת, חשובה יותר, היא הרצון להפריד בצורה לוגית בין משימות תכנות שונות. למשל: במקרה של מטלה בתוכנית שאינה מבוצעת לעיתים קרובות, הכללתה בטופס נפרד נראית יותר הגיונית מדחיסתה עם שאר פריטי התוכנית. בנוסף, ניתן לחסוך במשאבי המערכת בעת הרצת התוכנית, על ידי טעינת טפסים רק כשהם נדרשים ופריקתם לאחר מכן. במילים אחרות התוכנית משתמשת בנפח זיכרון קטן ככל האפשר בעת ריצתה.

הוספת טפסים חדשים לתוכנית

כאשר Visual Basic מתחילה פרויקט חדש, הוא מעלה בדרך כלל טופס ריק אחד, כפי שמודגם בתרשים 11.5. עם תכנון הפרויקט מוסיף המפתח פקדים לטופס, וכותב קודים לטיפול באירועים הקורים בטופס זה.



תרשים 11.5: Visual Basic פותחת פרויקט חדש בטופס ריק יחיד

בשלב מסוים יכול המפתח להחליט, אם לשם טיפול במטלה חדשה, או לצורך הקלת הצפיפות בטופס ההתחלתי, שנחוץ לו טופס או טפסים נוספים. תהליך הוספת טופס הוא פשוט. ניתן ללחוץ על הלחצן Add Form, או לבחור Project, Add Form. פעולה זו מציבה טופס ריק חדש על המסך. טופס זה נראה כפי שנראה הטופס הראשון עם תחילת העבודה. אם לא ניתן שם לטופס הראשון, נקבע שם המחדל שלו Form1, הטופס השני נקרא Form2 (או Form3, Form4 וכן הלאה). אם ניתן לטופס הראשון שם שונה, ברירת המחדל לשם הטופס השני היא Form1.

טיפ:

ניתן להוסיף קבצים, טפסים, או מודולים מתפריט מקוצר, המופעל בלחיצה ימנית בתוך חלון Project.

לאחר הוספת הטופס החדש, ניתן להציב בו פקדים ולכתוב קוד בשביל האירועים שלו, ממש כפי שנעשה בטופס ההתחלתי. בנוסף דרוש לאפשר גישה לטופס החדש מטפסים אחרים של התוכנית. הטיפול בגישה כזאת הוא דרך שורות הקוד Load וכן Unload, והשיטות Show ו-Hide של אובייקט הטופס.

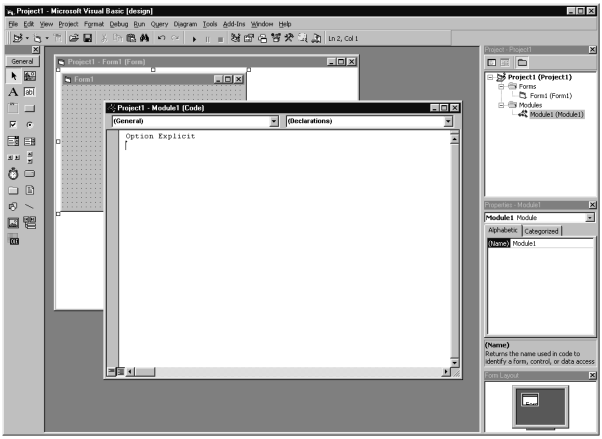
הוספת מודולי קוד לפרויקט

כשכותבים מספר קודים לטיפול באירועים ובמטלות נוספים, מגלים לעיתים קרובות שלאותה שיגרה נדרשת גישה ממספר מקומות בטופס או מטפסים שונים. במקרה כזה הגיוני לאחסן את השיגרה בקובץ מודול.

טיפ:

אם קיימת ספריה של פונקציות משותפות, לדוגמה תת-שגרות להדפסה, כדאי לשמור אותן במודול נפרד, כדי שניתן יהיה לצרפן לספריה של פרויקטים שונים.

שוב, קובץ מודול כולל רק קוד של Visual Basic. אין בו פקדים, גרפיקה או מידע חזותי אחר. אם רוצים להוסיף קובץ מודול שיאחסן שגרות, ניתן לעשות זאת או בלחיצת החץ שליד הלחצן Add Form ובחירת Module מהתפריט הנפתח, או על ידי בחירת Project, Add Module. כל אחת מפעולות אלו מוסיפה מודול חדש לפרויקט, ומציבה את המשתמש בחלון Code שלו (ראה תרשים 11.6).

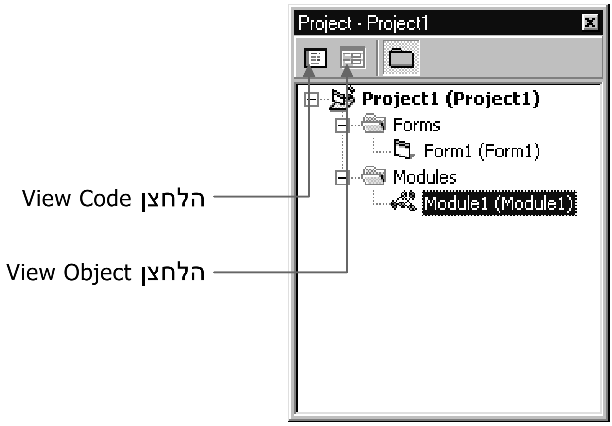


תרשים 11.6: ניתן לפתוח את החלון Code בלחיצה כפולה על שם המודול בחלון Project

כאשר מודול חדש נפתח לראשונה, נותנת לו Visual Basic את שם המחדל Module1 (או Module2 למודול השני וכן הלאה). כמו לגבי טפסים ופקדים, רצוי לתת גם למודולים שם ייחודי. למודול, כמו לטופס, יש מאפיין Name. לשינוי שם המודול, פשוט שנה את ערך המאפיין Name בחלון Property.

גישה לטפסים ולמודולים של פרויקט

עם הוספת טפסים ומודולים לתוכנית, הם מתווספים לחלון Project. חלון זה מאפשר גישה קלה לכל קטעי התוכנית (ראה תרשים 11.7). בחירת טופס או מודול נעשית בפשטות, על ידי לחיצה על שמם בחלון Project. ניתן אז ללחוץ בטופס על הלחצן View Object כדי לעבוד על תכנון הטופס, או ללחוץ על הלחצן View Code לעריכת הקוד הקשור לטופס. באשר למודולים, רק הלחצן View Code ניתן להפעלה, מכיון שלמודול אין אלמנטים חזותיים. לחיצה כפולה על שם הטופס תביא לתוצאה זהה לזו המושגת בלחיצה על הלחצן View Code.



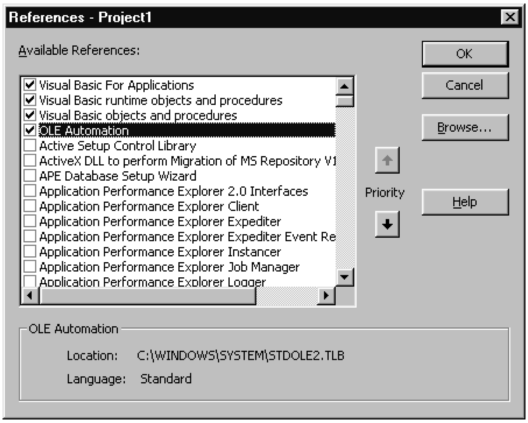
תרשים 11.7: החלון Project מאפשר גישה נוחה לכל הטפסים והמודולים

ניהול רכיבים בפרויקט

טפסים ומודולים הם רק שניים מסוגי הרכיבים שניתן להוסיפם לפרויקט. אפשר גם להוסיף **פקדים מותאמים אישית** (Custom Controls) וכן **מודולי מחלקה** (Class Modules). הקוד של אחדים מהרכיבים, כמו טפסים ומודולים, ניתן לעריכה. אחרים, כמו פקדים של ספק חיצוני ו**ספריות קישור דינמיות** (Dynamic Link Library, DLL) הם כבר אחרי הידור. למרות שסוגים אלה הם חלק מהפרויקט, אין הם מופיעים בחלון Project, ומוסיפים אותם באמצעות תיבות דו-שיח מיוחדות.

טיפול בהפניות תוכנית

אחד הגורמים הדורשים טיפול בתוכנית הם **הפניות** (References). הפניות מצביעות לעבר שגרות ספריה שונות, המאפשרות לקוד לבצע מטלות מוגדרות. למשל, אם מתכוונים לאפשר לתוכנית לגשת לבסיס נתונים, יש לציין שהתוכנית משתמשת בספריה Data Access Objects. ב- Visual Basic השליטה בהפניות פשוטה למדי. תיבת הדו-שיח References מאפשרת בחירה של הפניות הדרושות לתוכנית. לשם כך יש לסמן את תיבת הסימון המופיעה לצד ההפניה (ראה תרשים 11.8). סמן את אלו הדרושות לך, ובטל את הסימון מאלו שאינך זקוק להן. ניגשים לתיבת הדו-שיח על ידי בחירת Project, References.



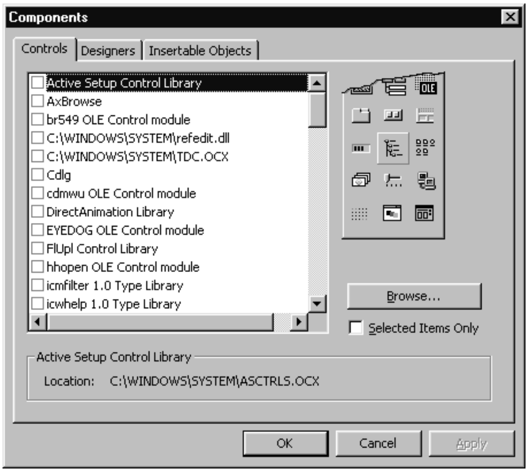
תרשים 11.8: תיבת הדו-שיח References מאפשרת בחירת ספריות שייעשה בהן שימוש בתוכנית

טיפ:

לאחר הוספת ההפניות לפרויקט ניתן לסקור את הקבועים והפונקציות הציבוריים שלו בסורק האובייקטים (Object Browser), הניתן לגישה בלחיצה על הלחצן Object Browser, או בהקשה על F2.

השליטה בפקדים

באופן דומה להפניות ספריה, ניתן להוסיף לפרויקט ולהסיר ממנו פקדים מותאמים אישית. כאשר Visual Basic נטענת למחשב, באופן אוטומטי מוטענים פקדים אחדים לארגז הכלים. אולם, בדרך כלל נדרשים פקדים שתוכננו לביצוע מטלות מסוימות, שהן מעבר ליכולתם של הפקדים הרגילים. ניתן לטפל בפקדים המיוחדים של פרויקט בשימוש בתיבת הדו שיח Components (ראה תרשים 11.9). כדי לגשת לתיבת דו-שיח זו בחר Project, Components. כמו בתיבת הדו-שיח References, ניתן לבחור את הפקדים המותאמים אישית שיתווספו לתוכנית בעזרת סימון תיבות סימון. עם היציאה מתיבת הדו-שיח, משתנה ארגז הכלים, ומציג את הפקדים החדשים.



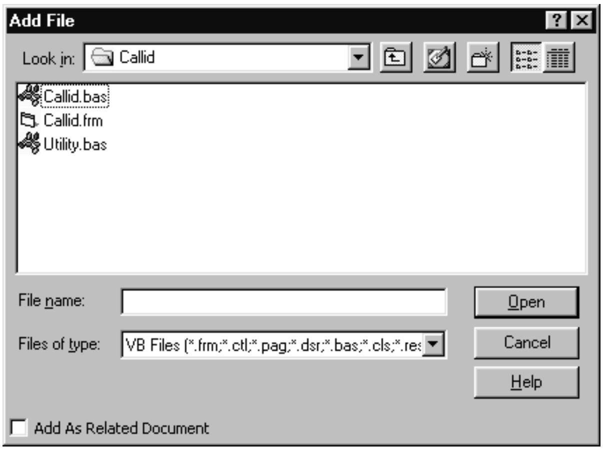
תרשים 11.9: תיבת הדו-שיח Components מאפשרת הוספת פקדים לפרויקט

הוספת טפסים, מודולים ומחלקות לפרויקט

מי שמפתח תוכניות מחשב, יגלה בשלב מסוים שבידיו שגרות וטפסים סטנדרטיים, שניתן להשתמש בהם בפרויקטים רבים. ייתכן גם שפותחו על ידו שגרות מותאמות לקליטת שמות וסיסמאות משתמשים, לפתיחת קבצים ולכמה פעולות נוספות, המבוצעות כמעט בכל תוכנית.

ניתן, לכל אחת ממטלות אלו ובכל תוכנית, לבנות מחדש את הטופס ולכתוב מחדש את השיגרה. אבל דרך זאת מאוד לא יעילה. דרך טובה יותר היא לעשות שימוש חוזר במודולים וטפסים שפותחו לפני כן ושכבר נוסו ונבדקו.

הכנסת מודולים וטפסים אלה לתוך הפרויקט הוא תהליך פשוט. בבחירת Project, Add File מביאים אל המסך את תיבת הדו-שיח Add File (ראה תרשים 11.10). תיבת דו-שיח זו מאפשרת לאתר ולבחור קבצים לשם הוספה לפרויקט הנוכחי. למרבה הצער, תיבת דו-שיח זו מאפשרת הוספת קובץ אחד בלבד בכל פעם. לכן, אם דרוש להוסיף יותר מקובץ אחד לתוכנית, יהיה צורך לחזור על הפעולה מספר פעמים.



תרשים 11.10: ניתן לגשת לתיבת הדו-שיח Add File ממערכת התפריטים, או מסרגל הכלים הסטנדרטי, וכן על ידי לחיצה ימנית בחלון Project או הקשת Ctrl+D

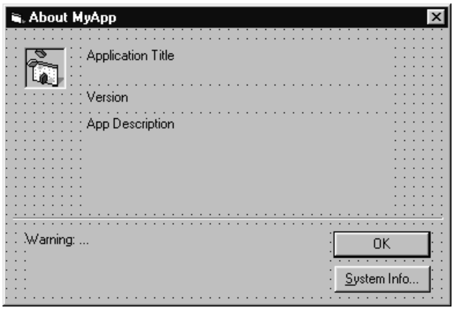
אזהרה:

כשמוסיפים את אותו טופס או מודול לפרויקטים נפרדים, יש לזכור כי שינוי פונקציות במודול משפיע על כל הפרויקטים המשתמשים בו. אם בכוונתך לשנות מודול משותף בצורה קיצונית, השתמש באופציה Save *modulename* As, הנמצאת בתפריט File, או העתק קודם את המודול לספרית משנה אחרת.

הערה:

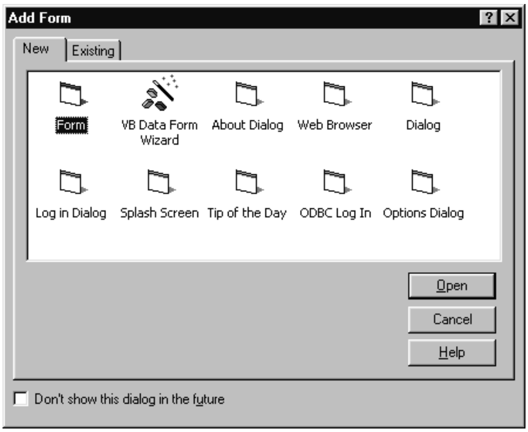
קבצים בעלי סיומת FRM. ו-FRX. הם קבצי טפסים. קבצים בעלי סיומת BAS. הם קבצי מודול.

אפשר גם להשתמש בתבניות הטפסים של Visual Basic. תבניות אלו הן טפסים שהוגדרו מראש, ובנויים לפונקציות ספציפיות, כמו למשל הטפסים About Box, Splash Screen, DataGrid וכן Tip of the Day. יש להוסיף לטופס את הגרפיקה המתאימה, תוויות כותרת ולכתוב קוד מינימלי, כדי להתאימו לצרכים. כדוגמה ראה בתרשים 11.11 את תבנית הטופס About Box.



תרשים 11.11: תבניות טפסים מקלות על פיתוח קטעי תוכנית משותפים

כדי לגשת לאחת מתבניות הטפסים, יש להציג את תיבת הדו-שיח Add Form בלחיצה על לחצן Add Form שבסרגל הכלים, או על ידי בחירת Project, Add Form. ניתן אז לבחור אחד מסוגי הטפסים בכרטיסיה New של תיבת הדו-שיח (ראה תרשים 11.12).



תרשים 11.12: ניתן בקלות להתאים תבניות טופס לשימוש בפרויקט

אם יצרת טופס שיהיה לו שימוש בתוכניות נוספות, ניתן גם לשמור אותו כתבנית. יש לשמור את הטופס בקובץ תבניות הטפסים של Visual Basic. בפעם הבאה כשתבקש להוסיף טופס חדש, תופיע גם התבנית החדשה בתיבת הדו שיח Add Form.

הערה:

אם Visual Basic מותקנת בתיקיית ברירת המחדל, תבניות הטפסים תהיינה מאוחסנות בקובץ Visual Basic\Templates\Forms.

הסרת קטעים

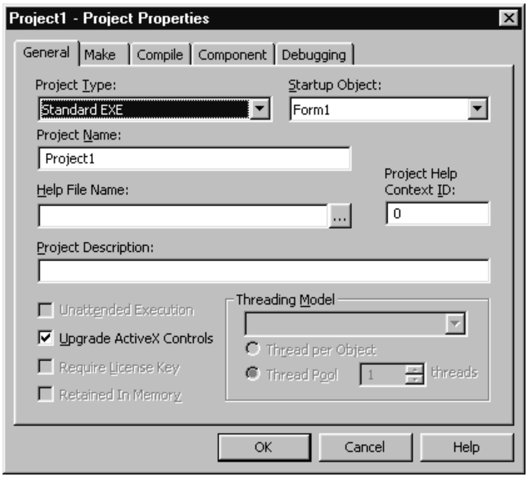
להסרת מודול או טופס מפרויקט, בוחרים את הפרויקט או המודול בחלון Project, ואז בוחרים Project, Remove. Visual Basic מבקשת אישור לכוונה להסיר את הקובץ, ואז מסירה אותו.

שליטה בהתחלת התוכנית

שמתחילים פרויקט תכנות, Visual Basic מניחה שהטופס הראשון שנוצר הוא זה שיוצג מייד בעליית התוכנית. למרות שבפרויקטים רבים זאת כוונת המפתח, יש במקרים רבים עניין לפתוח בטופס שנוצר בשלב יותר מאוחר בתהליך הפיתוח. במקרים אחדים אף רוצים שהתוכנית תיפתח בלי טופס בכלל.

קביעת טופס פתיחה

אם הטופס הראשון איננו זה שבו מתכוונים לפתוח את התוכנית, Visual Basic מאפשרת לבחור איזה מהטפסים יוצג ראשון על ידי התוכנית. בחירה זו נעשית בתיבת הדו-שיח Project Properties, כמוצג בתרשים 11.13. ניגשים לתיבת דו-שיח זו על ידי בחירת Project, Project Properties.



תרשים 11.13: רשימת האובייקטים Startup מאפשרת בחירת הטופס הנטען בפתיחת התוכנית

שם שיגרה Sub Main

רשימת האובייקטים Startup כוללת בנוסף לכל הטפסים שבפרויקט, גם את הביטוי Sub Main. שם שיגרה שמור זה מאפשר לתוכנית להיפתח ללא טופס התחלתי. אם בוחרים באפשרות זו, אחד מקבצי המודול חייב לכלול שיגרה בשם Main.

סיבה אחת להתחיל תוכנית באפשרות Sub Main יכולה להיות הצורך לבצע שגרת אתחול לפני העלאת טופס כלשהו. סיבה אחרת לכך יכולה להיות פיתוח תוכנית שירות לשורת פקודה, שאיננה דורשת אינטראקציה עם מפעיל.

מכאן...

פרק זה סיפק מבט על ניהול חלקי התוכנית השונים. הוצגו טכניקות אחדות לתכנות יעיל יותר. בנוסף, נחשף הקורא לרבים מהרכיבים המגוונים המרכיבים את התוכנית השלמה. מידע נוסף על יצירת החלקים השונים והשימוש בהם ניתן למצוא בפרקים הבאים:

- לדיון בסיסי כיצד לכתוב קוד של Visual Basic, ראה פרק 9 **יסודות התכנות של Visual Basic**.
- ללמוד יותר על הפיכת קוד ליעיל יותר, ראה פרק 10 **שליטה במהלך התוכנית**.

חלק 3

רכיבי תוכנית ב- Visual Basic



12

הפקדים המשותפים של Microsoft

מה בפרק?

- היכרות עם פקדים משותפים
- פקד ImageList - פקד משותף בסיסי
- ארגון הנתונים
- קבלת קלט מהמשתמש
- דיווח אודות מצב התוכנית והתקדמות התוכנית

ל- Visual Basic סידרה של **פקדים משותפים** (Common Controls) המאפשרים לפתח תוכניות הכוללות מאפיינים זהים למיגוון רחב של תוכנות, מתוצרת Microsoft ומתוצרת יצרנים אחרים. פקדים אלה יאפשרו לך ליצור **סרגלי כלים** (Toolbars) **ושורות מצב** (Status Bars) ויאפשרו להציג נתונים באופנים שונים. פרק זה ימשיך את הדיון בפקדים על ידי הצגת היכולות של הפקדים המשותפים של Microsoft.

מבוא לפקדים משותפים

הפקדים המשותפים של Windows (Windows Common Controls) הם קבוצת פקדים המאפשרים לשלב בתוכניות יכולות כלליות המוצעות על ידי סביבת Windows ומכאן השם "משותף". בשפת Visual Basic 6 חלק מהפקדים המשותפים הקיימים עודכנו ונוספו להם יכולות חדשות, כן נוספו מספר פקדים משותפים חדשים לגמרי.

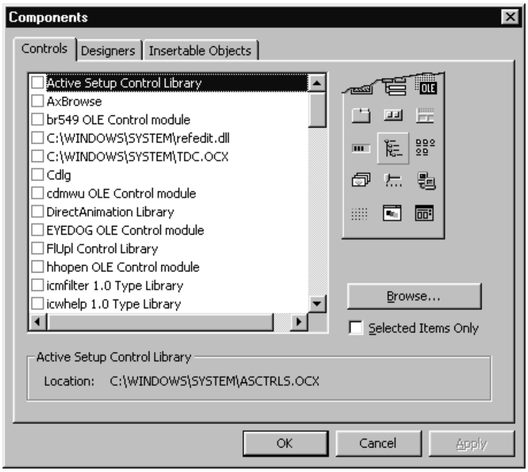
בטבלה 12.1 תובא רשימה מלאה של הפקדים המשותפים שמוצעים לך בשפת Visual Basic ובכללם פקדים הכלולים בארגז הכלים של Visual Basic.

טבלה 12.1: הפקדים המשותפים של Microsoft

שם הפקד	תיאור
ImageList	משמש לאחסון קבוצת תמונות המשומשות בפקדים אחרים
TabStrip	מאפשר לכלול בטפסים דפי מאפיינים בעלי מבנה כרטיסיות
ToolBar	מציג סרגל כלים גרפי במסגרת יישום (פקד זה תואר בפרק 6)
StatusBar	מציג נתונים בשורת המצב בדרך כלל בתחתית החלון
TreeView	מציג נתונים במבנה של עץ הניתן להרחבה ולצמצום, בדומה לחלונית השמאלית של סייר Windows
ListView	פקד מתקדם להצגת רשימות התומך בכמה מבטים, דומה לחלונית הימנית של סייר Windows
Slider	מציג פס מחוון, הדומה למחוונים של מערכת סטריאו
ImageCombo	מהווה שיפור של פקד ComboBox (תיבה משולבת) הרגיל שתואר בפרק 4. זהו פקד חדש שנכלל לראשונה בשפת Visual Basic 6
Animation	מאפשר שילוב קטעי וידאו חסרי קול בטפסים
UpDown	פקד הכולל לחצני חיצים המשמשים להגדלת והקטנת ערכים
MonthView	תומך בתצוגת לוח שנה חודשי, דומה לזה הכלול ב- Windows 98
DTPicker	מאפשר להזין תאריך ידנית או על ידי שימוש בפקד MonthView
CoolBar	פקד מכולה עבור סרגלי כלים הניתנים להסטה, כדוגמת אלה הכלולים בסביבת הפיתוח המשולבת של Visual Basic

לפני שיתאפשר לך להשתמש בפקדים שפורטו כאן תידרש להוסיף אותם לארגז הכלים שלך על ידי ביצוע הפעולות המפורטות להלן:

1. בחר את הפקודה **Components** מתפריט **Project**. תוצג לפניך תיבת הדו-שיח **Components**.
2. בחר את קבוצת הפקדים המשותפים שאותה תרצה להתקין (ראה תרשים 12.1). תוצענה לך שלוש הקבוצות המפורטות להלן:
   - **Microsoft Windows Common Controls 6.0** - כוללת את כל הפקדים שפורטו בטבלה 12.1 עד פקד ImageCombo ועד בכלל.
   - **Microsoft Windows Common Controls-2 6.0** - שאר הפקדים המפורטים בטבלה 12.1, פרט לפקד CoolBar.
   - **Microsoft Windows Common Controls-3 6.0** - פקד CoolBar.
3. לחץ על **OK** כדי לצרף את קבוצות הפקדים שבחרת לתיבת הכלים של Visual Basic (ראה תרשים 12.2).



תרשים 12.1: שלוש קבוצות הפקדים המשותפים סומנו והן תוספנה לתיבת הכלים לאחר שתלחץ על OK

אם תרצה להשתמש בפקדים המשותפים במסגרת פרויקט חדש, תידרש להוסיף אותם לארגז הכלים של Visual Basic. את פעולת ההוספה תידרש לבצע גם בכל מקרה בו תרצה להוסיף פקד משותף כלשהו לפרויקט קיים, שהפקדים המשותפים טרם הוספו אליו. לאחר שתוסיף אותם לארגז הכלים ותשמור את הפרויקט שלך, הפקדים המשותפים יוצגו בארגז הכלים של הפרויקט בפעם הבאה שתטען את הפרויקט.



תרשים 12.2: ארגז כלים אשר מכיל פקדים משותפים מכל שלוש הקבוצות

פקד ImageList: פקד משותף בסיסי

אופן פעולתו של פקד **ImageList** (רשימת תמונות) הוא כמשתמע משמו. פקד ImageList דומה לפקד תמונה או לפקד תיבת תמונה, אך הוא אינו משמש לאחסון תמונה יחידה אלא לאחסון אוסף תמונות. פקד ImageList גם אינו מכיל שום אלמנט המוצג בזמן ריצה, הוא משמש כשטח אחסון בלבד. אם תרצה להציג תמונה כלשהי המאוחסנת בפקד ImageList תידרש להסתייע בפקד נוסף.

חשוב שתבין את אופן פעולת הפקד, מפני שרבים מבין הפקדים המשותפים האחרים תלויים בתמונות המאוחסנות בפקד זה. למעשה, אופן השימוש העיקרי בפקד ImageList הוא לשם אחסון תמונות עבור פקדים אחרים, ורק לעיתים רחוקות הוא מתפקד כפקד עצמאי.

הגדרת פקד ImageList בעת עיצוב היישום

הדרך השכיחה ביותר להגדיר פקד ImageList היא על ידי צירוף תמונות באופן ידני בעת עיצוב היישום. אין צורך להפיץ את קבצי התמונות עצמם יחד עם היישום.

הערה:

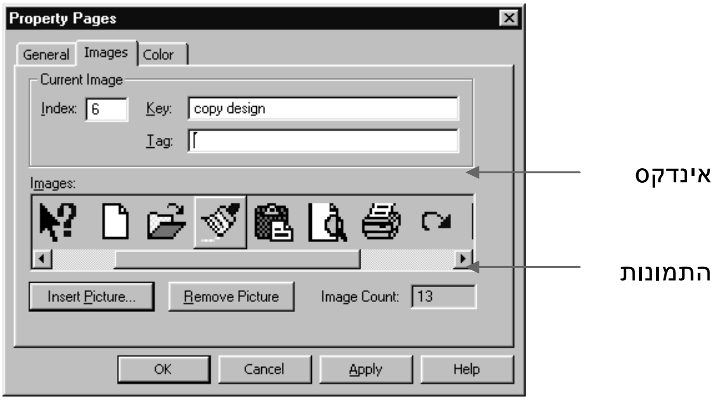
בפרק זה תמלא בפקדי ImageList קבצי תמונה שסופקו לך כחלק מתוכנת Visual Basic. את הקבצים הגרפיים האלה תמצא בתיקיית הקבצים הגרפיים של Visual Basic. תוכל גם למצוא את אותן תמונות (אם תחליט להתקין אותן) על ידי כך שתאתר את התיקיה המתאימה במחשב שלך. נוח מאוד להשתמש בסמלים (Icons) במסגרת פקדי ImageList מפני שהם קטנים ומפני שניתן להציג תצוגה מקדימה שלהם בתיבת הדו-שיח File.

בצע את הפעולות המפורטות להלן כדי להגדיר פקד ImageList בעת עיצוב היישום:

1. שרטט פקד ImageList על גבי הטופס שלך והגדר עבורו שם ייחודי.
2. פתח את תיבת הדו-שיח **Property Pages** של הפקד על ידי לחיצה ימנית על הפקד ובחירה בפקודה **Properties** מהתפריט המוקשר שיוצג לפניך.
3. בחר את כרטיסיית **Images** של תיבת הדו-שיח (ראה תרשים 12.3). הכרטיסיה תאפשר לטעון אלמנטים גרפיים ולהגדיר מזהה ייחודי עבור כל אחד מהם, כמתואר בסעיפים הבאים.
4. להוספת תמונה לפקד ImageList לחץ על לחצן **Insert Picture**. תוצג לפניך תיבת הדו-שיח **Select Picture**. בתיבת דו-שיח זו תוכל לבחור בקבצים מהסוגים Icon, Cursor, GIF ו-JPEG שיצורפו לרשימה.

כאשר תוסיף לרשימה תמונות, יוגדרו עבורן ערכי אינדקס באופן אוטומטי (ראה שוב את תרשים 12.3). תוכל גם להגדיר **מזהי מחרוזת** (String Identifiers) עבור תמונות על ידי כך שתקליד את המזהים בשדה **Key** של תיבת הדו-שיח. מומלץ להשתמש במפתחות מפני שכך יתאפשר לך להתייחס לכל תמונה בנפרד על ידי שימוש בשם ייחודי, ללא תלות בסדר התמונות ברשימה.

5. כאשר תסיים להוסיף תמונות לרשימה, לחץ על **אישור** כדי לסגור את תיבת הדו-שיח.



תרשים 12.3: אם תגדיר פקד ImageList במסגרת עיצוב היישום, התמונות תאוחסנה במסגרת הטופס ותהודרנה כחלק מקובץ הביצוע של היישום

הגדרת פקד ImageList באמצעות קוד

ניתן גם להגדיר ולשנות פקד ImageList בזמן ריצה, על ידי שימוש בקוד תוכנית. בעת גישה לפקד ImageList כל תמונה מטופלת כפריט ב**אוסף** (Collection).

הערה:

אוספים (Collections) הם מושג חשוב אשר משמש למספר מטרות בעת העבודה בשפת Visual Basic. אם אוספים הם מושג חדש עבורך המשך בקריאה, במושג אוספים נדון לעומק בהמשך הפרק.

אוסף הוא קבוצת אובייקטים. פקד ImageList משמש לעבודה עם אוסף בשם ListImages. האובייקטים המאוחסנים באוסף זה הם התמונות המאוחסנות בפקד ImageList. תוכל להשתמש בקוד לשם ביצוע פעולות על אוסף ListImages, כגון הוספת תמונות לאוסף ושינוי תמונות הכלולות באוסף. לדוגמה, שורת הקוד הבאה תצרף תמונה לפקד ImageList בשם ImageList1 :

```
ImageList1.ListImages.Add,,LoadPicture("D:\MYDIR\MYPIC.BMP")
```

ניתן לקרוא לשיטה Add של אוסף ListImages שוב כדי לצרף לאוסף כמה תמונות בעת עיצוב היישום. אוסף דומה במובנים מסוימים למערך אך אינו זהה לו. הגישה לאוסף מתבצעת כמו הגישה אל מערך, כפי שניתן לראות בדוגמת הקוד הבאה, אשר מציבה את התמונה הראשונה הכלולה בפקד ImageList במאפיין Picture של הטופס :

```
Set Form1.Picture = ImageList1.ListImages(1).Picture
```

בוודאי שמת לב שהגישה אל פריט מסוים באוסף מתבצעת על ידי שימוש ב**אינדקס** (Index), כאשר במקרה המוצג בדוגמה ערך האינדקס הוא 1. זכור שאוסף משמש לאחסון אובייקטים, בשונה ממערך שבו מאוחסנים ערכים מספריים. לכל אובייקט הכלול באוסף יש מאפיינים משלו.

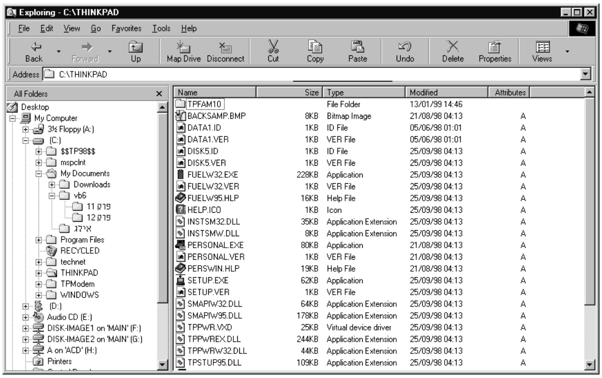
Key הוא מאפיין מיוחד המוגדר עבור אובייקטים הכלולים באוסף. מאפיין **Key** (מפתח) של אובייקט מכיל מחרוזת שניתן להשתמש באופן הדומה לשימוש באינדקס :

```
Set Form1.Picture = ImageList1.ListImages("Smiley Face").Picture
```

כדי להשתמש במאפיין Key לצורך חיפוש אובייקט, הצב ערך במאפיין זה של האובייקט בעת הוספת האובייקט לאוסף. עבור כל סוג של אוסף הוגדרו שיטות **Add** (הוספה), **Remove** (גריעה), שהמאפיינים Key וכן Index מהווים עבורן פרמטרים. אם מאפיין Index של אובייקט לא הוגדר, Visual Basic מציבה בו ערך באופן אוטומטי ואילו המאפיין Key הוא אופציונלי.

ארגון הנתונים

הפקדים ListView וכן TreeView מאפשרים לארגן נתונים לצורך הצגתם במבנה הדומה לזה שבמסגרת סייר Windows (ראה תרשים 12.4). היתרון העיקרי של פקדים אלה הוא העובדה שהם מאפשרים להפריד בין חזות הפקד עצמו, לבין הנתונים שמוצגים במסגרת הפקד. נתייחס כדוגמה לרשימה שמכילה 100 פריטים. פקד ListView יאפשר למיין את הפריטים ולשנות את המבנה בו הם יוצג על ידי שינויים בהגדרות המאפיינים של הפקד. זהו שיפור ניכר בהשוואה לפקד ListBox הרגיל, שאם תרצה לשנות את סדר הפריטים המוצגים במסגרתו, תידרש לטעון שוב את הפקד.



תרשים 12.4: לסייר Windows ממשק TreeView (חלונית שמאלית) **ו**ממשק ListView (חלונית ימנית)

בסעיף הבא תיצור פרויקט פשוט שיתבסס על פקד ListView ופרויקט פשוט שיתבסס על פקד - TreeView. כדאי שתהיה מודע לכך ששני הפקדים האלה הם מורכבים יחסית ללימוד. כדאי אולי שתקדיש קצת זמן לעיון בעזרה המקוונת, כדי לצבור מעט ידע אודות השימוש בהם. אם תקדיש מעט זמן למטרה זו, לא תתחרט.

שימוש בפקד ListView

פקד ListView דומה לפקד ListBox, אך הוא מכיל כמה שיפורים. פקד ListView יכול להציג נתונים במיגוון אופנים: רשימות המכילות סמלים שמסודרות משמאל לימין, רשימות במבנה טורי, ומבט דוח. אופני תצוגה אלה תואמים לאלה המוצעים בתפריט התצוגה של סייר Windows (סמלים גדולים, סמלים קטנים, רשימה ופרטים). לאחר שהפריטים יאוחסנו באוסף ListItems תוכל להיעזר במאפיין View של הפקד לצורך שינוי התצוגה למבנה הרצוי לך.

רשימה המוצגת במבנה דוח מיוחדת משום שהיא מאפשרת להציג יותר נתונים מכפי שאפשר להציג בעת שימוש במבני תצוגה אחרים. למשל, ב- Windows Explorer בחירה באפשרות **פרטים** (Details) תציג פרטים נוספים פרט לשם הקובץ. במבנה תצוגה כזה שם הקובץ יהיה הפריט הראשי ואילו גודל הקובץ, סוג הקובץ ותאריך היצירה של הקובץ, יהיו תת-פריטים המקושרים לשם הקובץ.

בעת שימוש בתוכניות תוכל לשלוט על תת-פריטים אלה על ידי שימוש במאפיין SubItems שמכיל מערך מחרוזות המקושרות אל כל פריט ברשימה. אוסף ColumnHeaders הוא שמפיק את כותרות הטורים שמוצגות בראש הדוח.

קודם לכן ציינו שפקד ImageList מספק את התמונות שהוצגו במסגרת פקדים משותפים אחרים. פקד ListView יכול להשתמש בעד שלושה פקדי ImageList מפני שהוא יכול לשמש להצגת סמלים גדולים, סמלים קטנים וכן תמונות במסגרת כותרות הטורים. את פקדי ImageList תוכל להגדיר בעזרת תיבת הדו-שיח Property Pages של פקד ListView או על ידי שימוש בקוד באופן הבא:

```
Set lViewMain.Icons = ImageList1
Set lViewMain.SmallIcons = ImageList2
Set lViewMain.ColumnHeaderIcons = ImageList3
```

היכולת לשילוב תמונות סמל בכותרות טורים היא יכולת חדשה שנכללה לראשונה בשפת Visual Basic 6. אם לא תגדיר את המאפיינים SmallIcons, Icons וכן ColumnHeaderIcons של פקד ListView, לא תוצגנה תמונות כלשהן במסגרת הפקד. כדאי להשתמש בפקד ImageList קטן יחסית (בגודל 16X16) בעת מילוי ImageList שתשמש להצגת סמלים קטנים.

בוודאי זכור לך שהפריטים הכלולים בפקד ImageList מאוחסנים במערך. כל הפריטים הכלולים בפקד מאוחסנים באוסף בשם ListItems ולכל פריט יש מספר מאפיינים, ביניהם אינדקס המצביע על התמונה המשויכת לפריט ו-Caption (כיתוב). בדומה לרוב האוספים גם עבור האוסף ListItems הוגדרה שיטת Add, שמוגדרת כך:

```
listView1.ListItems.Add, "mykey", "My Item", 1, 1
```

בעת בניית הפרויקט לדוגמה תלמד דברים נוספים על פרמטרים של השיטה Add של פקד ListView.

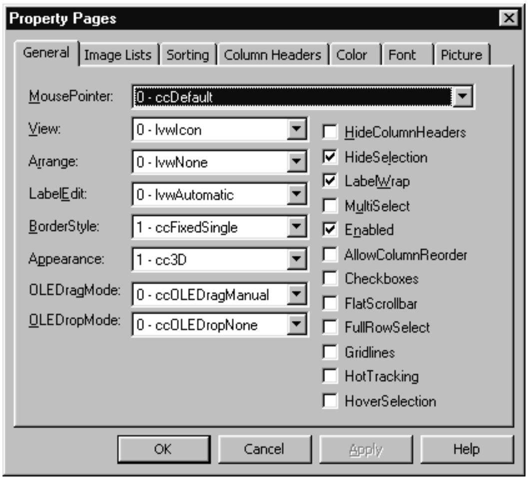
התחלת פרויקט הדוגמה

תוכנית הדוגמה תציג קבוצות כדור בסיס ואת נתוני הניצחונות וההפסדים שלהן. התחל את הפרויקט על ידי ביצוע הפעולות הבאות:

1. התחל פרויקט חדש מסוג **Standard EXE**.
2. הוסף לפרויקט שלך את קובץ הפקדים **Microsoft Windows Common Controls 6.0**, באופן שתואר בתחילת הפרק.
3. הוסף לטופס פקד **ListView** והגדר עבורו את השם **lvMain**.
4. הוסף לטופס שלושה פקדי **ImageList** והגדר עבורם את השמות **ilNormal**, **ilSmall**, ו-**ilHeader**.
5. הוסף את התמונות הדרושות לפקדי ImageList. הקוד לדוגמה שצורף לפרויקט מניח שהרשימות ilNormal, וכן ilSmall מכילות תמונה אחת בכל רשימה ואילו ilHeader מכילה שלוש תמונות.

הגדרת מאפיינים של פקד ListView

בדומה לפקדים אחרים, גם את המאפיינים של פקד ListView ניתן להגדיר בשני אופנים: באמצעות גיליון המאפיינים (Property Pages) של הפקד או על ידי שימוש בקוד. אם תרצה להגדיר את הפקד תוך כדי עבודה בסביבת העיצוב, תוכל להיעזר בתיבת הדו-שיח **Property Pages** אשר מוצגת בתרשים 12.5. לצורך פרויקט הדוגמה תגדיר 4 מאפיינים: View, Icon, SmallIcons ו-ColumnHeaderIcons. בסעיף זה תלמד כיצד להגדיר את המאפיינים האלה בעת העיצוב וכן יוצג לפניך קטע קוד שיגדיר את המאפיינים.



תרשים 12.5: תיבת הדו-שיח Property Pages תאפשר להגדיר מאפייני פקד ListView

תחילה תידרש להגדיר את מאפיין View, אשר מגדיר את המבנה בו הנתונים יוצגו במסגרת הרשימה. למאפיין זה ארבע הגדרות אפשריות:

- lvwIcon - מציגה כל פריט ברשימה כסמל גדול עם תיאור טקסטואלי פשוט.
- lvwSmallIcon - מציגה כל פריט כסמל קטן עם תיאור טקסטואלי פשוט. הפריטים מוצגים כשהם פרוסים אופקית.
- lvwList - מציגה את הסמלים בדומה לתצוגת Small Icons (ההגדרה lvwSmallIcon), פרט לכך שהסמלים מוצגים בפריסה אנכית.
- lvwReport - מציגה כל פריט במבנה הכולל סמל קטן, תיאור טקסטואלי ונתונים מפורטים על הפריט. גם הגדרה זו מציגה את הפריטים בטורים אנכיים.

בעת עבודה עם פקד ListView תידרש להגדיר גם את שמותיהם של פקדי ImageList שיכילו את הסמלים הקטנים, הסמלים הגדולים, והתמונות שתשולבנה במסגרת כותרות הטורים של הרשימה. אם תיעזר בתיבת הדו-שיח Property Pages תוכל להגדיר את שמות הפקדים באמצעות תיבות הרשימה הנפתחות שבכרטיסיה Image Lists של תיבת הדו-שיח. בתיבות הרשימה אלו תמצא את שמות כל פקדי ImageList שהוספת לטופס. ביישום הדוגמה תגדיר את מאפיין View באמצעות קוד.

הוספת פריטים לרשימה

הפעולה הבאה אותה תידרש לבצע היא כתיבת קוד שיצרף פריטים לתצוגת הרשימה. תוכל להשתמש בקטע הקוד לדוגמה שמוצג בתוכנית 12.1 כדי להוסיף פריטים לאוסף ListItems. הפעל את התוכנית ותוצג לפניך התצוגה הנראית בתרשים 12.6.

תוכנית 12.1: LISTVIEW.VBP - שימוש באירוע Load להגדרת פקד ListView

```
Private Sub Form_Load()

        'Set up the icons from the image List
         lvMain.Icons = ilNormal
         lvMain.SmallIcons = ilSmall
         lvMain.ColumnHeaderIcons = ilHeader


         ' Add ColumnHeaders. The width of the columns is the width
         ' of the control divided by the number of ColumnHeader objects.
         Dim clmX As ColumnHeader
         Set clmX = lvMain.ColumnHeaders.Add(,, "Team",,, 1)
         Set clmX = lvMain.ColumnHeaders.Add(,, "Wins",,, 2)
         Set clmX = lvMain.ColumnHeaders.Add(,, "Losses",,, 3)

         'Create a ListItem object.
         Dim itmX As ListItem

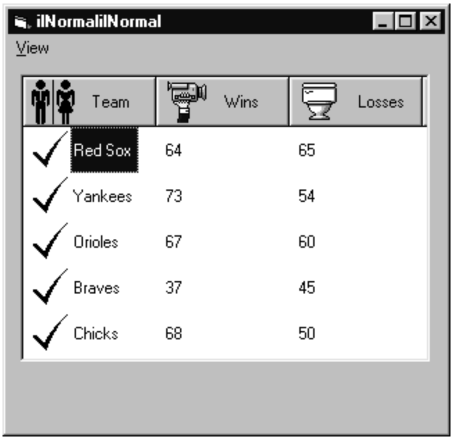
         'Add some data setting to the ListItem:

         'Red Sox Stats:
         Set itmX = lvMain.ListItems.Add(,, "Red Sox", 1, 1) ' Team
         itmX.SubItems(1) = "64" ' Wins
         itmX.SubItems(2) = "65" ' Losses

         ' You can duplicate the above 3 lines of code
         ' to add information for more teams…


         lvMain.View = lvwReport ' Set View property to Report.

End Sub
```



תרשים 12.6: כך ייראה פקד ListView שיוגדר על ידי קטע הקוד שהוצג כאן

שים לב בקטע הקוד שהובא כאן למשתני האובייקט הזמניים ששימשו לצורך הגדרת האוספים ColumnHeader וכן ListItems. תחילה קראת לשיטה Add אשר החזירה הפניה אל האובייקט שהוסף על-ידה. בקטע הקוד שהגדיר את כותרות הטורים זו היתה למעשה פעולה מיותרת, מפני שהקוד אינו מתייחס למשתנה הזמני clmX לאחר הקריאה הראשונה לשיטה Add.

אך כאשר הוספת את נתוני הקבוצות לא נדרשת להגדיר מפתח משלך. בוודאי זכור לך שמפתח הוא מחרוזת המהווה מזהה ייחודי של אובייקט הכלול באוסף. אם לא תגדיר מפתח משלך עבור אובייקט, Visual Basic תגדיר עבורך מפתח. על-כן נדרשת להיעזר באובייקט itmX לצורך הגדרת שני תתי-הפריטים. אם תגדיר מפתח תכתוב את קטע הקוד הרלוונטי כך:

```
lvMain.ListItems.Add, "RSox", "Red Sox", 1, 1
lvMain.ListItems("RSox").SubItems(1) = "64"
lvMain.ListItems("RSox").SubItems(2) = "65"
```

הוספת פריט ListItem לפקד ListView

הדוגמאות שבסעיף זה תצגנה את האוסף ListItems, שבו מאוחסנים פריטי ListItem שנוצרו על ידי שימוש בשיטה Add. תחביר השיטה Add של אוסף ListItem הוא:

```
object.add index, key, text, icon, smallicon
```

להלן פירוט הפרמטרים השונים הכלולים בפנייה לשיטה ושל משמעויותיהם:

- Object - האובייקט הקורא לשיטה. אלמנט הכרחי בשורת הקריאה.
- Index - מספר להגדרת מיקום פריט במסגרת האוסף. אם לא תציב ערך מספרי בארגומנט הזה, הפריט יתווסף לסוף הרשימה. הפרמטר Index הוא אופציונלי. ערך מאפיין index של פריט מסוים עלול להשתנות כתוצאה מהוספה או גריעה של פריטים אחרים.

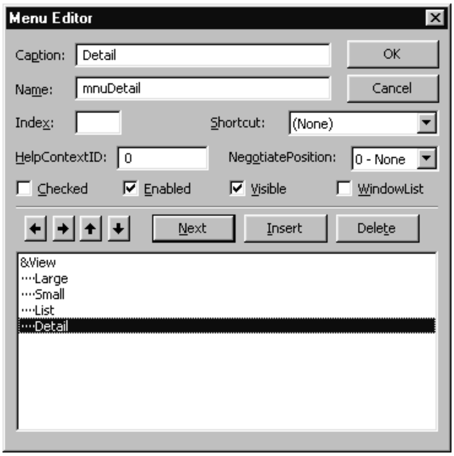
- Key - ניתן להשתמש במחרוזת זו כדי להגדיר לפריט שם ייחודי וידידותי. הפרמטר Key הוא אופציונלי.
- Text - פרמטר המכיל את המחרוזת שתרצה להציג במסגרת הפריט בחלון הפקד. זהו פרמטר אופציונלי, אך אני ממליץ למשתמשים בלתי מנוסים להשתמש בו. חשוב שלא לבלבל בין פרמטר זה ופרמטר Key.
- Icon - פרמטר המכיל את מספר האינדקס או המפתח של תמונה הכלולה בפקד ImageList ששמו הוצב במאפיין Icons של פקד ListView. פרמטר זה ישמש לבחירת התמונה הרצויה. זהו פרמטר אופציונלי, אך יש להתייחס אליו בזהירות מפני שאם לא תציב בו ערך כלשהו לא יתאפשר לך להציג סמלים גדולים בתצוגת lvwIcons של הפקד.
- SmallIcon - דומה לארגומנט Icon, אך מכיל את האינדקס של התמונה במסגרת פקד ImageList ששמו הוצב במאפיין SmallIcons של הפקד. זהו פרמטר אופציונלי, אך אם לא תציב בו ערך לא תתאפשר תצוגת סמלים קטנים במסגרת אופני התצוגה lvwSmallIcons, lvwList או lvwReport.

אם תרצה להוסיף לרשימה נתונים נוספים לאובייקט ListItem שיצרת, MyListItem, תידרש לשנות את הגדרת המאפיין SubItems(Index). מידע נוסף אודות מאפיין SubItems תוכל למצוא בעזרה המקוונת של Visual Basic.

שינוי התצוגה

לאחר שתציב נתונים בפקד ListView, תוכל להציג אותם במספר אופני תצוגה שונים, על ידי כך שתציב ערכים במאפיין View של הפקד תוך שימוש בקוד. אם תרצה לראות את מבני התצוגה השונים תוכל להוסיף תפריט View (תצוגה) ליישום הדוגמה.

צור את התפריט בעזרת Menu Editor (עורך התפריטים) של Visual Basic (לחץ על הלחצן Menu Editor או בחר Menu Editor מתפריט Tools). הפעל את Menu Editor וצור את פריטי התפריט המוצגים בתרשים 12.7.



תרשים 12.7: פריטי התפריט של תוכנית הדוגמה כשהם מוצגים בעורך התפריטים

לאחר שתסיים לעבוד עם עורך התפריטים, תידרש להוסיף קוד לכל אחד מפריטי התפריט שיצרת, כדי לאפשר לבצע את הפעולה הרצויה. הקוד הדרוש לפריטי התפריט השונים מוצג בתוכנית 12.2.

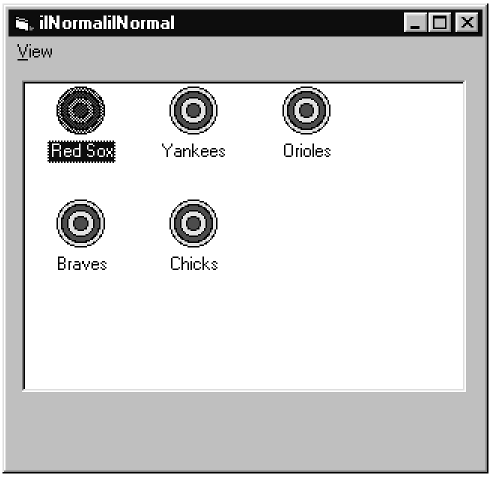
תוכנית 12.2: LISTVIEW.VBP - שימוש באירועי Menu Click עבור פרויקט הדוגמה

```
Private Sub mnuDetail_Click()
    lvMain.View = lvwReport
End Sub

Private Sub mnuLarge_Click()
    lvMain.View = lvwIcon
End Sub

Private Sub mnuList_Click()
    lvMain.View = lvwList
End Sub

Private Sub mnuSmall_Click()
    lvMain.View = lvwSmallIcon
End Sub
```



תרשים 12.8: בתצוגת Report מוצגים נתונים מפורטים אודות הפריטים, תוך שימוש בכותרות הטורים בעוד שבתצוגת Large Icons מוצגת רק תמונה המסמלת את הפריט מעל לשם הפריט

בשגרת האירוע Form_Load נוצרים אובייקטים של ListItem (קבוצות כדור בסיס) שיתוספו לאוסף ListItem. בכל פעם שפריט כלשהו מוסף לאוסף ListItems מוצבים ערכים בתת-הפריטים SubItems(1) (טור Wins - נצחונות) וכן SubItems(2) (טור Loss - הפסדים), של אובייקט ListItem, itmX.

שגרות אירוע Click של פריט התפריט מגדירות את מאפיין View של פקד ListView. הערכים שמאפיין View יכול לקבל הם lvwIcons, lvwSmallIcons, lvwList וכן lvwReport בהתאמה.

בתוכנית הדוגמה שהצגנו זה עתה הדגמנו את יכולתו של פקד ListView להציג פריטים. בדומה לפקד ListBox גם פקד ListView תומך באירועים רבים ובמאפיינים רבים, כדוגמת אירוע ItemClick המיועד לבחור פריטים ולבצע עליהם פעולות שונות.

אירוע ItemClick מעביר אובייקט אל התוכנית שלך, כדי לאפשר לך לבצע על האובייקט את פעולה המתאימה:

```
Private Sub ListView1_ItemClick(ByVal Item As ComctlLib.ListItem)
    If Item.Text = "Magic Item" Then MsgBox "You Clicked the magic item!"
End Sub
```

פקד ListView תומך גם במאפיינים המאפשרים למיין את תצוגת הדוח תוך שימוש בכותרות הטורים. השימוש באירוע ColumnClick מאפשר לך לשנות בקלות את סדר המיון של הטור הנבחר, כמתואר כאן:

```
Private sub lvMain_ColumnClick(ByVal ColumnHeader As ComctlLib.ColumnHeader)
    lvMain.SortKey = ColumnHeader.Index -1
    lvMain.Sorted = True
End Sub
```

ייתכן ששמת לב לכך שניתן לשנות שמות פריטים הכלולים בפקד ListView באמצעות לחיצה בודדת על טקסט הפריט. הפקד תומך בשני אירועים BeforeLabelEdit וכן AfterLabelEdit אשר מאפשרים לתוכנית שלך לדעת אילו פריטים משונים כעת.

נוכחת שפקד ListView הוא פקד מורכב יחסית. אם תרצה להתנסות בעבודה עם כל היכולות שלו ייתכן שכדאי שתבנה כמה תוכניות לדוגמה שתאפשרנה לך להתנסות בעבודה עם כל המאפיינים והשיטות שלו.

יכולות חדשות של פקד ListView

אם התנסית בשימוש בפקד ListView שנכלל בגרסאות קודמות של Visual Basic, תשמח בוודאי לשמוע שלפקד ListView שנכלל בגירסה 6 נוספו שיפורים רבים. כבר התנסית בשימוש באחד השיפורים האלה, האוסף ColumnHeaderIcons אשר מאפשר להציג תמונות במסגרת כותרות הטורים של תצוגת הדוח. להלן מפורטים כמה שיפורים נוספים אשר יאפשרו לך לשפר חזות פקד ListView:

- AllowColumnReorder - מאפיין זה מאפשר לשנות את סדר תצוגת הטורים בתצוגת lvwReport (תצוגת דוח). לדוגמה, אם תגדיר את המאפיין כ-True בפרויקט הדוגמה יתאפשר לך לגרור את טור Wins אל הקצה השמאלי של הדוח.
- FullRowSelect - מאפיין חדש זה הופך את פקד ListView למועמד מצוין להחליף את פקד ListBox. ייתכן שהבחנת שאופן הפעולה המהווה ברירת מחדל עבור

הפקד הוא לאפשר בחירת פריטים רק על ידי לחיצה על הטור השמאלי ביותר. הצבת ערך True במאפיין FullRowSelect מאפשרת לבחור פריטים בלחיצה על טור כלשהו בתצוגה וגם מאפשרת חיווי חזותי לכך, על ידי הצגת השורה המוארת על כל רוחב המסך.

- GridLines - מאפיין זה קובע האם יוצגו קווי רשת במסגרת הדוח. קווים כאלה יקנו לדוח מראה של גיליון נתונים.
- FlatScrollBars - מאפיין זה יציג פסי גלילה עם המראה החדש, השטוח.
- HoverSelection - במצב רגיל תידרש ללחוץ על פריט בפקד ListView כדי לבחור אותו. בהגדרת True למאפיין HoverSelection, הפריט ייבחר (ואירוע ItemClick יתבצע) גם אם תעביר את מצביע העכבר על גבי הפריט למשך מספר שניות.
- HotTracking - מאפיין אשר נועד להתאים את חזות הפקד למקובל בסביבת Active Desktop של Microsoft.
- CheckBoxes - מאפיין CheckBoxes קובע האם תוצגנה תיבות סימון לצד כל פריט שיוצג במסגרת פקד ListView. תוכל לזהות מהו מצב סימון התיבה או להגדיר את מצב הסימון על ידי שימוש במאפיין Checked של הפריט. אירוע ItemCheck יתבצע גם כאשר המשתמש יסמן את תיבת הסימון.

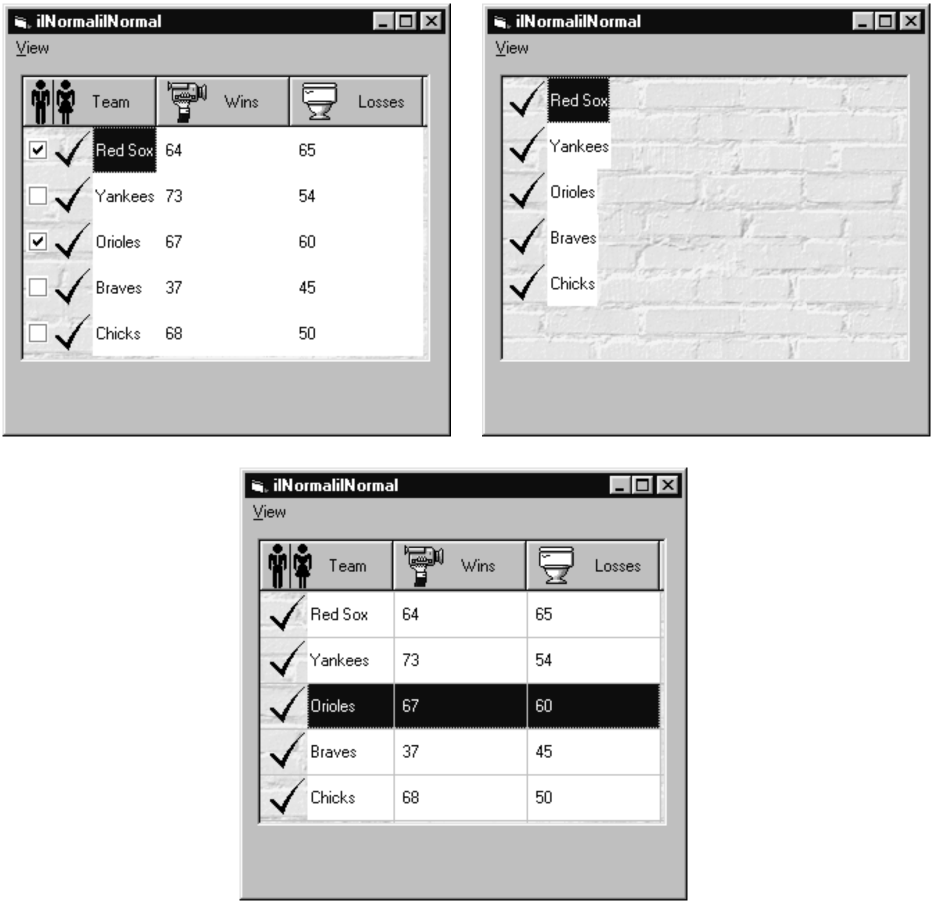
יכולת חדשה נוספת שנכללה לראשונה בפקד ListView של Visual Basic 6 היא יכולת להציג תמונה ברקע הפריטים. יכולת זו מיושמת כמו תצוגת רקע בעמוד Web. אם תטען תמונה על ידי הגדרת מאפיין Picture של פקד ListView, כמו בדוגמה הבאה:

```
ivMain.Picture=LoadPicture("D:\Pictures\Test.JPG")
```

בנוסף לכך, שני המאפיינים המפורטים להלן יגדירו את אופן הפעולה של התמונה ואת האינטראקציה שתתקיים בינה לבין הפריטים הכלולים ברשימה:

- מאפיין TextBackground קובע האם הטקסט יוצג על רקע שקוף או אטום.
- מאפיין PictureAlignment שולט על האופן בו התמונה תצויר על גבי פקד ListView. תוכל להתחיל את השרטוט במרכז הפקד או בכל אחת מפינות הפקד ולהציג חלק גדול ככל שניתן מתוך התמונה, או אם התמונה קטנה מהפקד תוכל להציג כמה תמונות שתהיינה פרוסות כמו אריחים.

בתרשים 12.9 תוכל לראות דוגמאות לשימוש בכמה מאפשרויות תצוגה אלו.

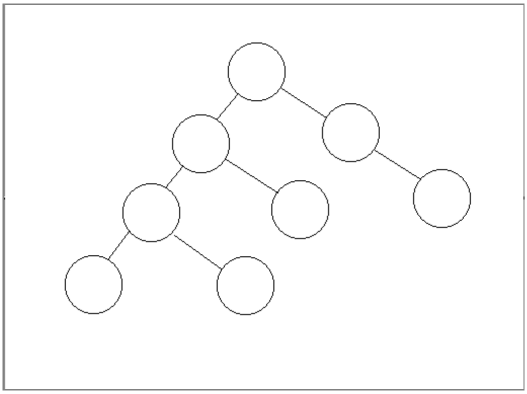


תרשים 12.9: לא ניתן היה להציג את מבני התצוגה המוצגים כאן על ידי שימוש בגרסאות הקודמות של פקד ListView

פקד TreeView

פקד TreeView דומה לפקד ListView אך הוא מסוגל להציג גם פריטים המכילים שילוב של טקסט וגרפיקה. פקד TreeView גם מציג את הפריטים הכלולים בו במבנה הירכי של עץ. אם למדת אי פעם קורס בסיס במתמטיקה או במדעי המחשב הרי שהמושג "מבנה הירכי של עץ" בוודאי מוכר לך. בתרשים 12.10 מוצגת דוגמת מבנה הירכי של עץ.

לאור אופיו ההירכי של פקד ListView יש כמה מושגים בסיסיים שהכרחי להבין אותם כדי לנצלו באופן יעיל: **שורש** (Root), **אב** (Parent), **וצאצא** (Child). פקד TreeView גם עושה שימוש נרחב באובייקט **Node**, כך שבכדי שתוכל להשתמש ביעילות באובייקט Treeview תידרש גם להיות בקי בשימוש בסוג אובייקט זה. הדוגמה השכיחה ביותר לשימוש בפקד TreeView היא התצוגה בחלונית השמאלית של סייר Windows (ראה תרשים 12.4).



תרשים 12.10: מבנה עץ מגדיר מערכת קשרים בין אובייקטים (מיוצגים על ידי עיגולים)

הבנת המושג Node (צומת)

מבנה הירכי הוא מבנה שלכל פריט הכלול בו יש קשר מוגדר עם הפריטים האחרים. נתייחס כעת אל אילן היוחסין של משפחתך כאל דוגמה של עץ. ההורים שלך מצויים "מעליך" במבנה ההירכי של המשפחה והצאצאים שלך מצויים "מתחתייך" במבנה.

בדוגמת אילן יוחסין של משפחה, כל אדם הוא **צומת** (Node) בעץ. הקשרים בין צמתים מיוצגים על ידי **ענפים** (Branches). הענפים היוצאים מצומת מסוים מקשרים את הצומת אל צמתים אחרים בעץ (כלומר ענפי העץ מקשרים אותך אל אבותיך), אך ייתכן שלצומת מסוים לא יהיו צאצאים.

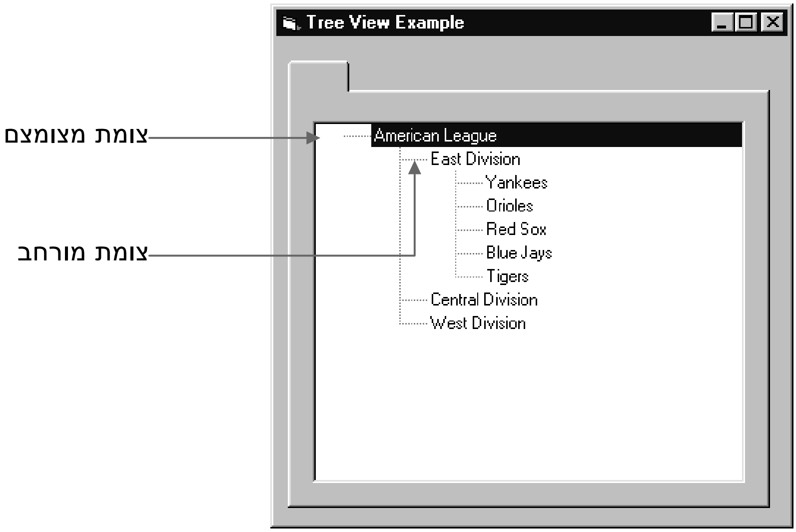
בדומה לעצי יוחסין ולעצים הקיימים בטבע, כל פקדי TreeView כוללים צמתים. מאפיין Nodes של פקד TreeView הוא בעצמו אוסף אובייקטים מסוג Node. אם עקבת אחר הדיון במושג אוסף שהובא עד כה, הרי שלא תתקשה להבין את משמעות המשפט האחרון (אם פרט כלשהו הקשור במושג אוסף אינו ברור לך כדאי שתעיין שוב בסעיפים שעסקו בפקד ListView). בדומה לאוסף ListItems של פקד ListView גם אוסף Nodes של פקד TreeView תומך במאפיינים מסוימים ובשיטות מסוימות, שביניהן השיטה Add אשר משמשת ליצירת אובייקט Node חדש:

```
tv1.Nodes.Add,, "mykey", "Test Node", 1, 2
```

בדומה לשיטות אחרות שאותן תיארנו, גם שיטה זו מחזירה הפניה אל האובייקט שיצרה זה עתה, כמתואר בקטע הקוד הבא:

```
Dim tempNode As Node
Set tempNode = tv1.Nodes.Add(,, "mykey", "Test Node")
tempNode.Image = 1
tempNode.SelectedImage = 2
tempNode.ExpandedImage = 3
```

פקד TreeView מאפשר להציג את העץ במבנה הניתן להרחבה ולצמצום, כמתואר בתרשים 12.11.



תרשים 12.11: מבנה העץ הפשוט המוצג כאן נוצר על ידי שימוש בפקד TreeView

משתמשים יכולים לשנות את העץ באמצעות העכבר, תוך הרחבה וצמצום ענפים בהתאם לרצונם. הבעיה העיקרית העומדת בפני מפתח המשתמש בפקד TreeView היא הצגת הצמתים במבנה הרצוי למשתמש והבהרת הקשרים הקיימים בין הצמתים.

על ידי שימוש במאפיין SingleSel תוכל לגרום לכך שהעץ ירחיב ענף באופן אוטומטי בעקבות בחירה בו.

הבנת מאפיין Root

בראש העץ נמצא שורש העץ (בניגוד למקובל בעצים המצויים בטבע). השורש העץ הוא צומת שממנו נובעים כל הצמתים האחרים הכלולים בעץ. לכל עץ יש רק שורש אחד. העץ המוצג בתרשים 12.11 הוא תרשים ארגוני פשוט של חברה קטנה. הצומת המכונה **American League** הוא שורש העץ, מפני שזהו הצומת הראשון שצורף לפקד TreeView.

לכל צומת הכלול באוסף Nodes יש מאפיין Root אשר מצביע על הצומת המהווה את שורש העץ. אם תרצה לבדוק שכל הצמתים הכלולים בעץ המוצג בתרשים 12.11 אכן מצביעים על אותו שורש תוכל לעשות זאת באמצעות קטע הקוד הבא:

```
Dim tempNode As Node
For Each tempNode in tv1.Nodes
	Print tempNode.Text & "'s root is " & tempNode.Root.Text
Next tempNode
```

חשוב לזכור ש-Root הוא צומת בפני עצמו ובכל צומת אחר בעץ מצויה הפניה אליו במאפיין ששמו Root.

עבודה עם מאפיין Parent

זה עתה ראית שלכל צומת הכלול בעץ יש הפניה לשורש העץ. אך הכרת שורש העץ אינה מספיקה להגדרת מיקום צומת כלשהו במסגרת המבנה ההירככי של העץ.

אם תרצה לדעת את מיקומך במסגרת המבנה ההירככי המוצג בתרשים 12.11 יידרשו לך נתונים נוספים בנושא למיקום השורש. כעת אתה מוכן ללמוד על הראשון מבין כמה סוגי יחסים הקיימים במסגרת עץ, **יחס הורות** (Parent Relationship). בכדי שצומת מסוים יהיה הורה, נדרש שיהיו לו צאצאים. בתרשים 12.11 הצומת הראשון הוא ההורה של שני הצמתים שאחריו. לשלושת צמתי **Division** המוצגים בתרשים יש הורה משותף.

נניח שאתה צומת קטן ושאפתן, אשר נוקט בכל האמצעים הדרושים לשם טיפוס במעלה הסולם. תוכל להיעזר בקטע הקוד הבא כדי לבחון את מיקומך הנוכחי:

```
Dim MyNode As Node
Set MyNode = tv1.Nodes("Me")
If MyNode Is MyNode.Root Then
      MsgBox "You're the boss!"
Else
      MsgBox "Try getting promoted to " & MyNode.Parent.Text
End If
```

בדוגמה תוכל לראות שלכל צומת יש מאפיין Parent אשר מצביע על צומת ההורה שלו. היוצא מן הכלל היחיד הוא צומת Root, שמאפיין Parent שלו אינו מצביע על שום גורם אחר (במאפיין הזה מוצב הערך המיוחד Nothing).

עבודה עם מאפיין Children

אם תרצה לבדוק אם צומת כלשהו הוא אב לצמתים אחרים תוכל לבדוק מהו הערך המאוחסן במאפיין Children שלו, על ידי שימוש בקטע הקוד הבא:

```
Private Sub TreeView_NodeClick(ByVal MyNode As Node)
      If MyNode.Children = 0 Then
            MsgBox "I am Not a parent"
      End If
End Sub
```

ניתן לראות שמאפיין Children יחזיר מספר שלם אשר מייצג את מספר הצאצאים שיש לאותו צומת. בכדי שצומת ייחשב לצאצא של צומת אחר הוא חייב להיות קשור אליו בקשר ישיר.

עבודה עם המאפיין Child

המאפיין Children מחזיר את מספר הצאצאים שיש לאותו צומת ואילו המאפיין Child מחזיר את אחד הצמתים שהוא צאצא של הצומת (בדומה למאפיינים Root ו-Parent). אך במקרה שלאותו צומת יש מספר צאצאים, מאפיין Child שלו מחזיר רק את הצאצא הראשון שלו. מאפיין Child של צומת שאין לו צאצאים מכיל ערך Nothing, על-כן לא כדאי לך לנסות לבצע גישה אל מאפיין Child של צומת שמאפיין Children שלו מכיל את הערך 0.

השאלה המתבקשת היא כיצד ניתן לגשת אל צאצאים של צמתים שהם אינם הצאצא הראשון של צומת האב שלהם. לכל צומת יש כמה מאפיינים אשר מקלים על זיהוי האחים והאחיות שלו, כפי שניתן לראות בדוגמת הקוד הבאה:

```
Sub PrintAllChildren( )
    Dim anyNode As Node
    Dim kidNode As Node
    Dim inCounter As Integer
    For Each anyNode In tv1.Nodes
        If anyNode.Children <> 0 Then 'this node is a parent
            Print anyNode.Text & "'s Children are: "
            Set kidNode = anyNode.Child
            Print kidNode.Text

            inCounter = kidNode.FirstSibling.index
            While inCounter <> kidNode.LastSibling.Index
                Print tv1.Nodes(inCounter).Next.Text
                inCounter = tv1.Nodes(inCounter).Next.Index
            Wend
        End If
    Next anyNode
End Sub
```

זכור שמאפיין Child של צומת יכול להצביע רק על צומת אחד ואילו כל שאר הצמתים שחולקים את אותו אב נחשבים לצומתי **Next** או **Previous**. אך לכל הצמתים שלהם צומת אב משותף יש גם מאפייני FirstSibling וכן LastSibling.

הערה:

עבודה עם אובייקט Node עלולה להיות מלווה בקשיים מסוימים. הדרך הטובה ביותר ללמוד לעבוד עם אובייקטים אלה היא על ידי עיון בקוד קיים המשמש לביצוע פעולות על אובייקט Node. אם כבר יש לך הבנה מספיקה של מושגים הקשורים במבנים הירככיים, תוכל להיעזר בעזרה המקוונת של Visual Basic. אני ממליץ לתרגל את השימוש בפקד TreeView.

אם תרצה ללמוד עוד על פקד TreeView המשך לקרוא פרק זה. סעיף זה התמקד בטיפול באובייקט Node שכבר כלול בפקד TreeView. בסעיף הבא תלמד ותתרגל הוספת אובייקט Node חדש לפקד, כחלק מפרויקט לדוגמה.

פקד TabStrip

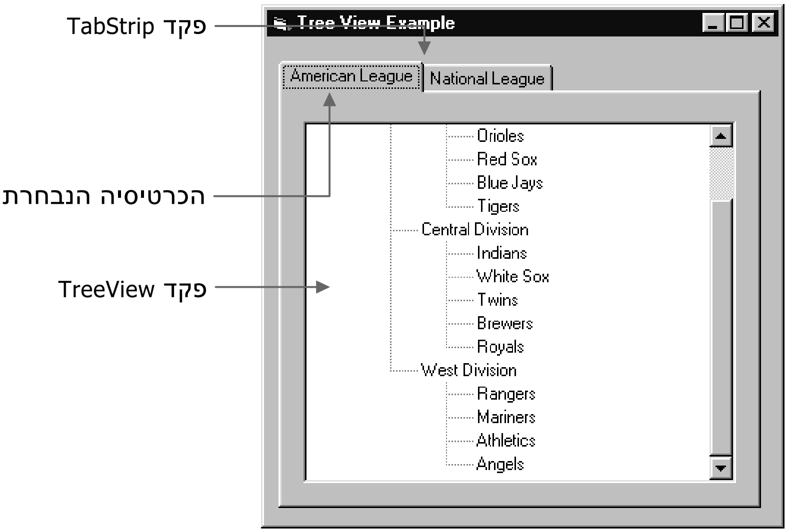
פקד TabStrip אשר מוצג בתרשים 12.12 הוא פקד שימושי מאוד הכלול באוסף הפקדים המשותפים של Windows. הוא מאפשר לחלק את הטופס מבחינה חזותית למספר כרטיסיות.

פקד TabStrip המוצג בתרשים מוצג כחלק מפרויקט הדוגמה שניצור במהלך סעיף זה. בפרויקט דוגמה זה נשתמש בפקד TabStrip לשם מעבר בין שתי רשימות. הרשימות שתוצגנה תהיינה רשימות של קבוצות כדור בסיס בשתי הליגות המקצועניות בארצות הברית שתוצגנה במסגרת פקדי TreeView. פקד TabStrip יאפשר לך לבחור את הליגה ולהציג את רשימת הקבוצות המשחקות בה.

התחלת פרויקט הדוגמה

התחל את פרויקט TreeTab על ידי יצירת פרויקט חדש מסוג Standard EXE. לאחר מכן בצע את הפעולות הבאות:

1. הוסף לטופס הפרויקט פקד **TabStrip** והגדר עבורו שם קצר אך בעל משמעות כגון **tsMain**.
2. הוסף לקובץ פקד **TreeView** והגדר את שמו **tv1**. כעת תהיה מוכן להתחיל בהגדרת מאפייני הפקדים



תרשים 12.12: פרויקט הדוגמה ייעזר בפקדי TabStrip ו-TreeView

אזהרה:

הפקד TabStrip אינו פקד מכולה (Container Control). כך שפקדים שישורטטו בתחומו לא יקבלו באופן אוטומטי את הגדרות המאפיינים Visible, Enabled, Top ו-Left שלו כפי שהיה קורה למשל עם פקדים שהיו משורטטים במסגרת פקד Frame. לחיצה על פקד TabStrip מביאה אותו לחזית התצוגה תוך הסתרת הפקדים המשורטטים עליו.

אם תרצה לשנות את מבנה התצוגה של פקדים שישורטטו במסגרת פקד TabStrip, כדאי שתקדיש תשומת לב לשיטה ZOrder של פקד TabStrip ושל פקדים אחרים העשויים להיות מושפעים משינוי סדר התצוגה. בעת שתעבוד במצב עיצוב ייתכן שיהיה עליך ללחוץ לחיצה ימנית על פקד TabStrip ולבחור את הפקודה Send To Back.

הגדרת פקד TabStrip

הצעד הבא בתהליך יצירת הפרויקט הוא יצירת הכרטיסיות שתיכללנה בפקד TabStrip. משום שתידרש להציג את רשימות הקבוצות בשתי ליגות, תידרשנה שתי כרטיסיות. הגדר את פקד TabStrip על ידי ביצוע הפעולות המפורטות להלן:

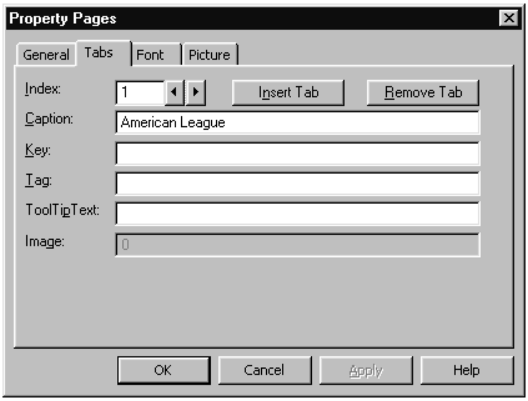
1. לחץ לחיצה ימנית על הפקד **tsMain** כדי להציג את התפריט המקוצר.
2. בחר מהתפריט את הפקודה **Properties** (הפקודה התחתונה בתפריט), כדי להציג את תיבת הדו-שיח **Property Pages** של הפקד tsMain.
3. לחץ על הכרטיסיה **Tabs** והקלד **American League** במאפיין **Caption**.
4. לחץ על לחצן **Insert Tab** כדי להוסיף כרטיסיה לפקד.
5. הקלד **National League** בתיבת הטקסט **Caption**.
6. לחץ על לחצן **OK** הכיתובים שהגדרת יוצגו במסגרת פקד TabStrip.

הגדרת פקד TreeView

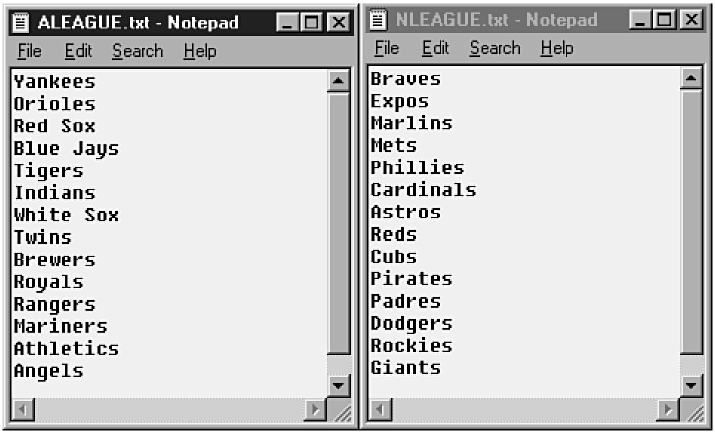
כפי שעשית בעת העבודה על פרויקט ListView גם כעת תידרש להגדיר את פקד TreeView על ידי שימוש בקוד לצורך צירוף אובייקטים לאוסף. תיעזר בפקד TreeView לצורך הצגת המבנה ההיררכי של הליגה שאותה תבחר להציג. בכל אחת מהליגות יש שלושה בתים: East, West ו-Central. הבתים כוללים חמש קבוצות, חמש קבוצות וארבע קבוצות, בהתאמה.

תוכל לנצל את המבנה האחיד של הליגות לתועלתך בכך שתאחסן את הנתונים של הקבוצות בשני קבצי טקסט, ALEAGUE.TXT ו-NLEAGUE.TXT שכל אחד מהם יכיל את שמות 14 הקבוצות שבאותה ליגה. את הקבצים תוכל ליצור בעזרת **פנקס הרשימות** (Notepad) של Windows או על ידי שימוש בכל עורך טקסט אחר שבו תעדיף להשתמש.

צור את השגרות המוצגות בתוכנית 12.13, אשר תוספנה את הנתונים המאוחסנים בקבצי הטקסט לפקד TreeView.



תרשים 12.13: על ידי שימוש בתיבת הדו-שיח Property Pages (שהיא עצמה מבוססת על פקד TabStrip) תוכל להוסיף בקלות כרטיסיות לפקד TabStrip



תרשים 12.14: בתוכנית הדוגמה פקד TreeView מלא בפריטי מידע שנקראים משני קבצי טקסט, כמתואר בתרשים

תוכנית 12.3: Treeview.VBP שימוש בשיגרה DisplayLeague לטעינת נתונים נבחרים אל תוך פקד TreeView

```
Public Sub DisplayLeague(sLeague As String)

    Dim tempNode As Node
    Dim nCounter As Integer
    Dim sTemp As String

    tv1.Nodes.Clear

    'Add the League
    Set tempNode = tv1.Nodes.Add(,, "R", sLeague)
```

```
    'Add the Divisions
    Set tempNode = tv1.Nodes.Add("R", tvwChild, "E", "East Division")
    Set tempNode = tv1.Nodes.Add("R", tvwChild, "C", "Central Division")
    Set tempNode = tv1.Nodes.Add("R", tvwChild, "W", "West Division")

    'Open the text file with team names
    If sLeague = "National League" Then
        Open App.Path & "\NLEAGUE.TXT" For Input As #1
    Else
        Open App.Path & "\ALEAGUE.TXT" For Input As #1
    End If

    'Add the 5 EAST teams
    For nCounter = 1 To 5
        Line Input #1, sTemp
        Set tempNode = tv1.Nodes.Add("E", tvwChild, "E" & nCounter, sTemp)
    Next nCounter
    tempNode.EnsureVisible
    'Add the 5 CENTRAL teams
    For nCounter = 1 To 5
        Line Input #1, sTemp
        Set tempNode = tv1.Nodes.Add("C", tvwChild, "C" & nCounter, sTemp)
    Next nCounter
    tempNode.EnsureVisible
    'Add the 4 WEST teams
    For nCounter = 1 To 4
        Line Input #1, sTemp
        Set tempNode = tv1.Nodes.Add("W", tvwChild, "W" & nCounter, sTemp)
    Next nCounter
    tempNode.EnsureVisible

    'Close the Input file
    Close #1

    'Set the Desired style
    tv1.Style = tvwTreelinesText
    tv1.BorderStyle = vbFixedSingle

    'Set the TreeView control on top
    tv1.ZOrder 0
End Sub
```

שים לב שקטעי קוד אלה מגדירים את הקשרים שיתקיימו במסגרת פקד TreeView בעת הוספת אובייקט Node. הפרמטר הראשון של השיטה Add הוא מציין של צומת שהוא "קרוב משפחה" של הצומת שקרא לשיטה, ואילו הפרמטר השני מציין את קרבת המשפחה שקיימת בין הצומת שקרא לשיטה לבין הצומת שנוסף על ידי השיטה. הקבוע tvwChild מציין שהצומת החדש הוא צאצא של הצומת שקרא לשיטה.

ביצוע פעולות במסגרת התוכנית

כעת נשוב ונעסוק בפקד TabStrip ונגדיר את שגרת Load של הטופס. סביר להניח כי תרצה שכאשר הטופס יוצג לראשונה יהיה בו מידע כלשהו, על כן תבצע קריאה לשיגרה DisplayLeague כדי להציג את רשימת הקבוצות שב- American League. אך הלחיצה על פקד TabStrip לא תשפיע על פקד TreeView מפני שטרם הוספת את הקוד הדרוש לשגרת האירוע Click של פקד TabStrip. קטע הקוד המוצג בתוכנית 12.4 קורא לשיגרה DisplayLeague תוך העברת שם הליגה הרצויה כפרמטר השיגרה.

תוכנית 12.4: Treeview.VBP - עבודה עם האירועים Load ו-Click

```
Private Sub Form_Load()

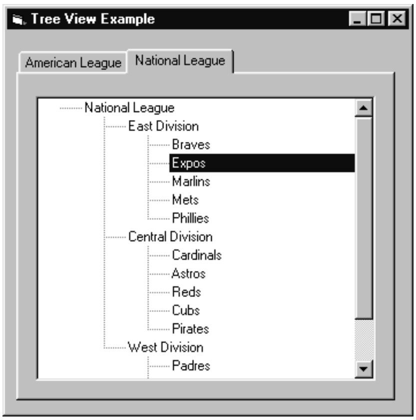
    'Put the tree view on top of the tab
    tv1.ZOrder 0

    'Call our procedure
    DisplayLeague "American League"
End Sub

Private Sub tsMain_Click()
    Dim nTemp As Integer

    nTemp = tsMain.SelectedItem.Index
    DisplayLeague tsMain.Tabs(nTemp).Caption

End Sub
```



תרשים 12.15: בלחיצה על כרטיסיה, יישום הדוגמה מציג רשימת קבוצות כדור בסיס

השיגרה DisplayLeague היא לב הפרויקט. שיגרה זו יוצרת תחילה צמתים שמכילים את שמות הבתים שבכל ליגה:

```
Set tempNode = tv1.Nodes.Add("R", tvwChild, "E", "East Division")
```

הקבוע tvwChild מורה ליישום שהצומת החדש שייווצר יהיה צאצא של הצומת "R".

לאחר מכן מוסיפים לכל בית מספר צמתים של קבוצות:

```
Set tempNode = tv1.Nodes.Add("E", tvwChild, "E" & nCounter, sTemp)
```

הארגומנט הראשון של השיטה Add הוא "E" כלומר ערך המפתח הייחודי של הבית המזרחי של הליגה. המפתח הזה מאפשר לצומת "E1" (הצומת המייצג את קבוצת Yankees) לדעת שהוא צאצא של הצומת שמייצג את הבית המזרחי.

כעת תורה לצומת של הבית ולכל הצאצאים שלו להשתמש באופן התצוגה המורחב (אובייקטי Node הכלולים בפקד TreeView יכולים להיות מוצגים כשהם מורחבים או כשהם מצומצמים), על ידי שימוש בשורת הקוד הבאה:

```
MyNode.EnsureVisible
```

(לחילופין ניתן גם להציב ערך True במאפיין Expanded של כל אחד מהצמתים המייצג בית בליגה).

הנושא האחרון לו נידרש להקדיש תשומת לב הוא האופן בו פקד TabStrip מודיע שבוצעה לחיצה על אחד מאובייקטי Node הכלולים בו. פקד TabStrip מכיל מערך אובייקטי Tab (כרטיסיות). כך שכאשר אתה לוחץ על כרטיסיה אתה למעשה בוחר כרטיסיה, באופן דומה לבחירת פריט IndexList בעת שימוש בפקד ListBox. הידע הזה יאפשר לך להבין טוב יותר מדוע שורת הקוד המוצגת להלן משמשת לזיהוי הכרטיסיה שעליה לחץ המשתמש:

```
nTemp = tsMain.SelectedItem.Index
```

בדוגמה הזו אחזרת את ערך האינדקס של הכרטיסיה שעליה לחץ המשתמש ואחסנת אותו במשתנה nTemp.

קבלת קלט מהמשתמש

מספר פקדים משותפים של Windows מציעים דרכים שונות לקבלת קלט מהמשתמש. לדוגמה, השימוש בלוח שנה להזנת תאריכים אינה הכרחית, אך היא תעניק לתוכנית שלך חזות מקצועית. אחד היתרונות החשובים של הפקדים המשותפים של Visual Basic הוא שהם מאפשרים לך ליצור במהירות תוכניות בעלות חזות מרשימה, על ידי שימוש באובייקטים קיימים. בסעיף זה נתייחס לפקדים הבאים:

- **ImageCombo** - תיבה משולבת גרפית.
- **MonthView** - פקד לוח שנה.

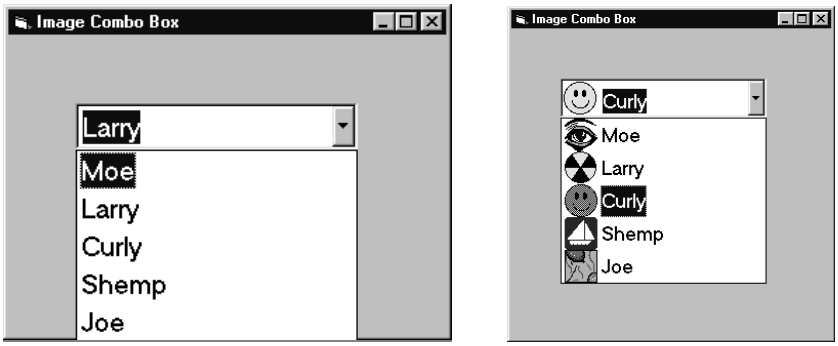
❖ **DTPicker** - תיבה משולבת לבחירת תאריכים.

❖ **UpDown** - משמש להעשרה של טיפול בקלט מספרי.

❖ **Slider** - משמש להזנה של מספרים באופן גרפי.

פקד ImageCombo

פקד ImageCombo הוא אחד הפקדים החדשים בגירסה 6 של Visual Basic. תפקידו דומה לזה של ComboBox שתואר בפרק 4. אך פקד ImageCombo מאפשר לשלב תמונות לצד כל אחד מהפריטים המוצגים במסגרת הרשימה הנפתחת של התיבה המשולבת, כפי שניתן לראות בתרשים 12.16.



תרשים 12.16: מראה פקד ImageCombo מרשים יותר ממראה פקד ComboBox רגיל

הערה:

הבדל נוסף הקיים בין הפקדים הוא העובדה שפקד ImageCombo תומך רק בסגנון תצוגה אחד, סגנון הרשימה הנפתחת ואילו פקד ComboBox מציע כמה סגנונות תצוגה.

הגדרת פקד ImageCombo

בדומה לפקדים משותפים אחרים של Windows גם פקד ImageCombo משתמש בפקד ImageList לאחסון התמונות שתוצגנה במסגרת התיבה המשולבת. לכל פריט הכלול באוסף ComboItems של התיבה המשולבת יש שתי תמונות שניתן להגדיר עבורם אינדקסים:

❖ Image - הצב במאפיין את אינדקס התמונה הרצויה להצגה משמאל לפריט.

❖ SelImage - זהו מאפיין אופציונלי. אם תציב בו ערך התמונה שתוצג לצד הפריט תשתנה בעקבות בחירה או עקב העברה של מצביע העכבר עליה. אם לא תציב ערך במאפיין, הפקד ישתמש בתמונה המוגדרת במאפיין Image.

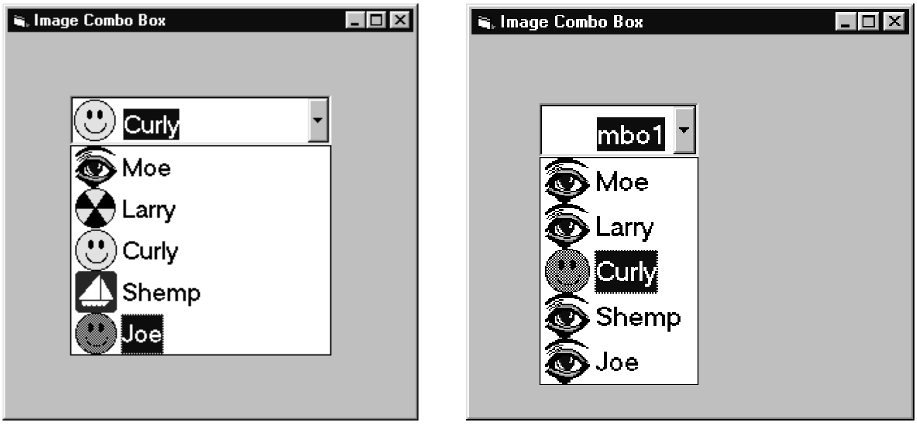
כדי ללמוד כיצד לקיים אינטראקציה עם פקד ImageCombo בצע את הפעולות הבאות:

1. התחל פרויקט חדש מסוג Standard EXE והוסף לו פקד **ImageList**.
2. הגדר את גודל התמונות שתוצגנה במסגרת הפקד ל-16X16 והוסף לפקד שתי תמונות.
3. הוסף לטופס פקד ImageCombo והוסף את שורות הקוד הבאות לשגרת אירוע Load של הטופס:

```
ImageCombo1.ImageList = ImageList1
ImageCombo1.ComboItems.Add,, "Moe" 1, 2
ImageCombo1.ComboItems.Add,, "Larry" 1, 2
ImageCombo1.ComboItems.Add,, "Curly" 1, 2
ImageCombo1.ComboItems.Add,, "Shemp" 1, 2
ImageCombo1.ComboItems.Add,, "Joe" 1, 2
```

שים לב לכך שפקודות Add משתמשות באותם ערכי אינדקס עבור מאפייני Image ו-SelImage של כל הפריטים.

4. הפעל את התוכנית והעבר את מצביע העכבר על גבי כל אחד מהפריטים המוצגים ברשימה. כאשר תאיר פריט מסוים התמונה המוצגת לצדו תשתנה. בנוסף לכך, אם תבחר את הפריט, תוצג לצדו התמונה שעליה מצביע מאפיין SelItem של הפריט, כמתואר בתרשים 12.17.



תרשים 12.17: ניתן להשתמש במאפיין SelImage לצורך הבדלה בין פריטים מוארים ונבחרים לבין פריטים רגילים

עבודה עם הפריט הנבחר

כל פריט המוצג בפקד ImageCombo כלול באוסף ComboItems של הפקד. אם תרצה לזהות האם המשתמש בחר פריטים כלשהם, בדוק האם מאפיין SlectedItem של הפקד מכיל הפניות אל פריטים כלשהם, על ידי שימוש בקטע הקוד הבא:

```
If ImageCombo1.SelectedItem Is Nothing Then
	' No Item is Selected!
Else
	MsgBox "You Selected: " & ImageCombo1.SelectedItem.Text
End If
```

הערה:

ניתן גם לבחור פריטים מקוד על ידי הצבת ערך True במאפיין Selected של הפריט, כמו בדוגמה הבאה:

```
ImageCombo1.ComboItems(1).Selected = True
```

מדוע יש צורך לבדוק את מאפיין SelectedItem כדי לראות האם הוא מכיל ערך Nothing? סביר להניח שזכור לך שבעת שימוש בפקד ComboBox משתמשים יכולים לא רק לבחור פריטים מרשימת הפריטים של הפקד, אלא יכולים גם להקליד טקסט בשורת הטקסט של הפקד. גם פקד ImageCombo תומך באופן פעולה זה. נסיון להקליד בשורת הקלט טקסט שאינו מייצג פריט, יגרום לשגיאה.

הערה:

אתה יכול לאלץ את המשתמשים לבחור את אחד הפריטים שיוצגו ברשימת הפריטים של הפקד, על ידי כך שתגדיר את מאפיין Locked של הפקד לערך True. הגדרה כזו תנטרל את היכולת להזין טקסט במסגרת הפקד.

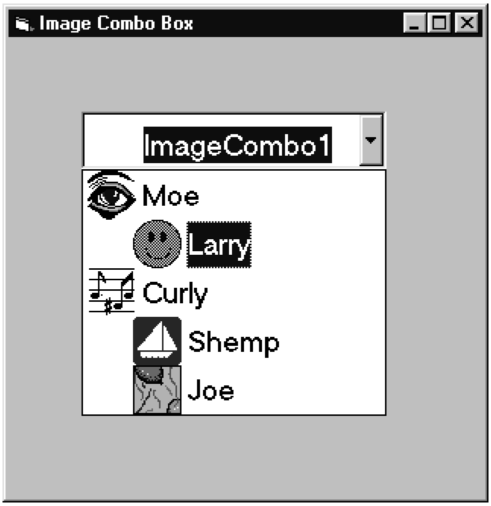
סידור פריטים בתיבה המשולבת

פקד ImageCombo גם מאפשר להגדיר רמת **הזחה** (Indentation) שונה עבור כל פריט הכלול באוסף ComboItems שלו. על ידי כך שתגדיר אינדקסים עבור פריטים שונים תוכל להעניק למשתמשי הפקד שלך תחושה של סדר וארגון.

להגדרת רמת ההזחה של פריט הצב במאפיין Indentation שלו ערך שלם כלשהו. את המאפיין ניתן להגדיר גם על ידי הכללת פרמטר אופציונלי נוסף במסגרת הקריאה לשיטה Add, כמתואר בדוגמה הבאה:

```
cbMain.ComboItems.Add "mykey", "mtext", 1, 2, 1
```

כל רמת הזחה מייצגת 10 פיקסלים. בתרשים 12.18 מוצג פקד ImageCombo פשוט שבמסגרת הגדרתו נעשה שימוש במאפיין Indentation.



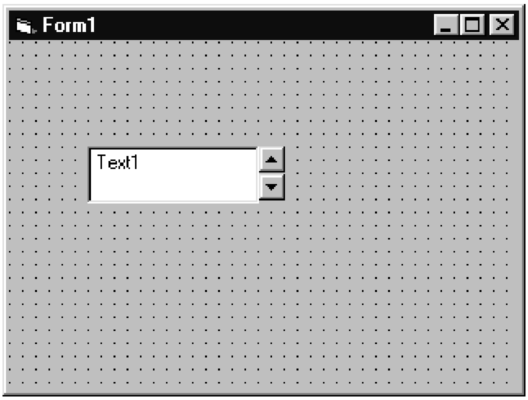
תרשים 12.18: ניתן להסתייע במאפיין Indentation לצורך ארגון פקד ImageCombo

בתקליטור:

דוגמה מלאה לפקד ImageCombo תמצא בתקליטור בפרק 12 בתיקיה ImageCombo.

פקד UpDown

פקד UpDown הוא פקד משותף נוסף אשר משמש לקבלת קלט מהמשתמש. אך פקד זה הוא מיוחד משום שאינו מיועד לשימוש בפני עצמו. במקום זאת הוא פועל בשילוב עם פקד המהווה עבורו "בן-זוג". פקד UpDown מהווה אמצעי המקל על שינוי הערכים המספריים המאוחסנים במסגרת הפקד האחר. לדוגמה, אם כלולה בטופס שלך תיבת טקסט שאמורה להכיל ערך מספרי, תוכל לקשר אותה אל פקד UpDown כדי לאפשר למשתמשים לעבור על פני מספרים מבלי להידרש להקליד אותם.



תרשים 12.19: פקד UpDown מאפשר למשתמשים להתאים ערכים לצרכיהם

הגדרת פקד UpDown

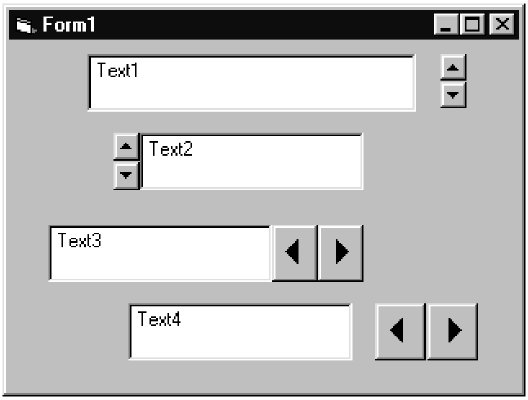
בצע את הפעולות הבאות כדי להגדיר פקד UpDown:

1. שרטט על הטופס פקד שיכול להכיל ערכים מספריים, כגון פקד TextBox.
2. שרטט על הטופס פקד **UpDown**.
3. הצב את שמו של הפקד שישמש להצגת הערכים במאפיין BuddyControl של פקד UpDown. הגדר את המאפיין על ידי הקלדת שם הפקד (לדוגמה Text1) או על ידי שימוש במאפיין AutoBuddy.
4. הגדר את מאפיין BuddyProperty של פקד UpDown כ-Default. הצבת ערך זה במאפיין תגרום לפקד UpDown לעדכן את מאפיין ברירת המחדל של פקד Buddy שלו (למשל, מאפיין ברירת המחדל של פקד TextBox הוא Text). תוכל גם להשתמש בפקד לשם עדכון מאפיינים אחרים, כל עוד אלה יהיו מאפיינים המאפשרים אחסון ערכים מספריים.

לאחר שתסיים להגדיר את פקד UpDown תוכל להגדיר מאפיינים נוספים שיגדירו את חזות הפקד ואת אופן פעולתו. שני המאפיינים שיאפשרו להגדיר את חזות הפקד הם:

- Alignment - קובע האם פקד UpDown יוצג מימין לפקד Buddy או משמאלו.
- Orientation - קובע האם הלחצנים של הפקד יוצגו אופקית או אנכית.

תוצאות השימוש במאפיינים Alignment ו-Orientation מוצגות בתרשים 12.20.



תרשים 12.20: ניתן להציב את פקד UpDown בכמה פריסות שונות

ניתן לומר שהמאפיינים המגדירים את אופן פעולת הפקד הם חשובים יותר עבור מפתחים. מאפיינים אלה הם:

- Increment - קובע שיעור שינוי בערך המספרי שיתרחש בעקבות לחיצה על לחצן.
- Max - מגדיר את הערך המקסימלי שהפקד יכול להכיל.
- Min - מגדיר את הערך המינימלי שהפקד יכול להכיל.

- SyncBuddy - גורם לפקד UpDown לעדכן את פקד Buddy שלו. ייתכן שתרצה להציב ערך False במאפיין אם אתה משתמש בפקד UpDown מבלי לקשר אותו לפקד. דוגמה לשימוש בפקד UpDown ללא קישור לפקד אחר הוא לצורך מעבר בין עמודי מסמך.
- Wrap - קובע האם לאחר לחיצה על לחצן ההגדלה כאשר בפקד מוצג הערך המקסימלי, יוצג בו הערך המינימלי (ולהפך).

עבודה עם ערכים ואירועים

בעת עבודה עם פקד UpDown תוכל לזהות מהו המספר שנבחר על ידי המשתמש באחד משני אופנים: על ידי אחזור הערך המאוחסן במאפיין Value של פקד UpDown, או במאפיין המתאים של פקד Buddy שלו. ברוב המקרים אין משמעות באיזה מבין שני הערכים האלה תבחר להשתמש. אך במקרים בהם מתאפשר למשתמשים לאחסן בפקד Buddy ערכים החורגים מהתחום שהוגדר עבור פקד UpDown, תוכל להיעזר רק במאפיין המתאים של פקד Buddy.



אזהרה:

חשוב שתזכור שמאפיין SyncBuddy שתואר קודם לכן מעדכן את פקד Buddy בערכים המוצבים בו, אך לא להפך. כלומר שאם נתייחס לרגע לדוגמאות שהוצגו בתרשימים, אם משתמש יזין ערך באחת מתיבות הטקסט, מאפיין Value של פקד UpDown לא יתעדכן. דרך אפשרית לפתרון הבעיה היא על ידי עדכון של מאפיין Value של תיבת הטקסט במסגרת אירוע Change של פקד Buddy שלו.

בנוסף לאירועים ה"רגילים" שמוגדרים עבור רוב הפקדים, פקד UpDown תומך גם בשלושה אירועים נוספים העשויים לעניין אותך:

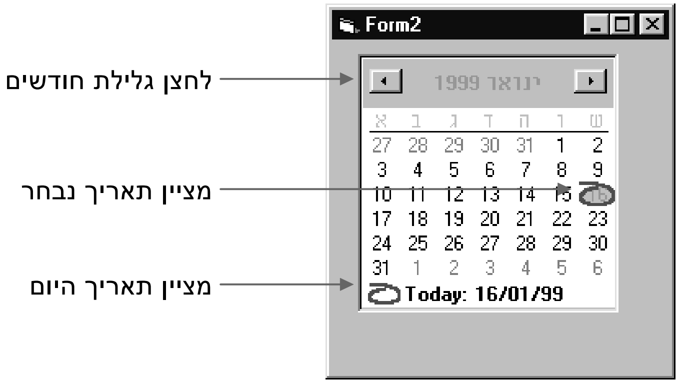
- Change - מתרחש בכל פעם שבה משתנה הגדרת המאפיין Value של הפקד.
- DownClick - מתרחש כאשר המשתמש לוחץ על החץ למטה של הפקד.
- UpClick - מתרחש כאשר המשתמש לוחץ על החץ למעלה של הפקד.

עבודה עם תאריכים

ב- Visual Basic 6.0 נוספו לקבוצת הפקדים המשותפים השנייה (הפריט Microsoft Windows Common Controls-2 6.0 בתיבת הדו-שיח Components) שני פקדים אשר מקלים מאוד על הזנת נתוני תאריך, פקד MonthView ופקד DTPicker. פקד MonthView שמוצג בתרשים 12.21 הוא פקד לוח שנה רב-גוני אשר מאפשר למשתמשים לבחור תאריכים באמצעות תצוגה חזותית במקום להקליד אותם.

שינוי חזות פקד MonthView

תרשים 12.21 מציג את המראה ברירת המחדל של פקד MonthView. אך לפקד יש כמה מאפיינים שתוכל להגדיר אותם כדי לשנות דברים בחזותו או באופן פעולתו. חלק ממאפיינים אלה מפורטים בטבלאות 12.2 ו-12.3



תרשים 12.21: פקד MonthView מאפשר למשתמשים לבחור ערכי תאריכים על ידי הצגת לוח שנה במקום על ידי הקלדה

טבלה 12.2: מאפיינים המגדירים את חזות פקד MonthView

שם המאפיין	ערך ברירת מחדל	תיאור
ShowToday	True	כן/לא תוצג בתחתית הפקד שורה עם תאריך היום
ShowWeekNumbers	False	כן/לא יוצג טור שבו יפורטו מספרי השבועות
ScrollRate	1	מגדיר כמה חודשים אחורה/קדימה התצוגה תעבור בעקבות הגלילה. גם אם במאפיין יוצב ערך הגדול מ-1 אפשר יהיה לעבור חודש אחר חודש על ידי לחיצה על התאריכים המוצגים בפינות הפקד
DayBold( )	False	קובע האם תאריך יוצג בכתב מודגש. מועיל להבלטת תאריכים מסויימים, כגון חגים. כדי להיעזר במאפיין זה תידרש לציין תאריך בקריאה למאפיין, כבדוגמה הבאה: MonthView1.DayBold(#7/4/1998#) = True

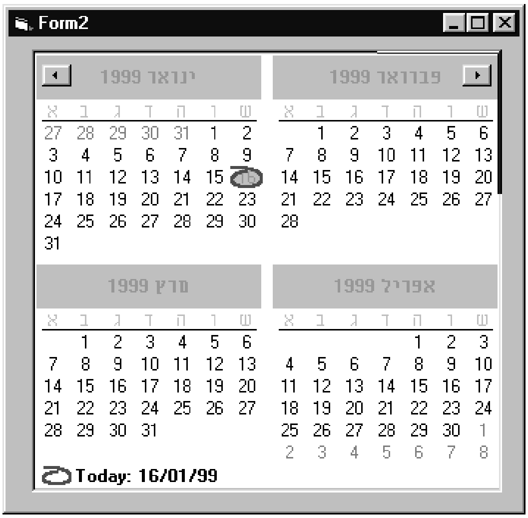
פקד MonthView מאפשר להגדיר גם צבעים שונים שבהם יוצגו הקטעים השונים הכלולים בו.

טבלה 12.3: מאפייני הצבע של פקד MonthView

שם	תיאור
ForeColor	קובע את צבע הימים והקו האופקי שמתחת לתוויות השבועות
TrailingForeColor	קובע את הצבע בו יוצגו הימים שאינם כלולים בחודש המוצג כעת אך נראים על המסך (הימים האחרונים של החודש הקודם והימים הראשונים של החודש הבא). הימים הראשונים של החודש הבא מוצגים לרוב כשטח ריק בסוף תצוגת החודש
MonthBackColor	מגדיר את צבע הרקע של ימי החודש. לא רקע כותרת הפקד
TitleBackColor	מגדיר את הצבע בו ייצבע הרקע של אזור הכותרת
TitleForeColor	מגדיר צבע טקסט שלה כיתוב הכותרת והתאריך הנבחר

הצגת מספר חודשים בבת אחת

אם תרצה להציג יותר מחודש אחד ברגע נתון במסגרת פקד MonthView, תוכל לעשות זאת על ידי הגדרת המאפיינים MonthRows ו-MonthsCols. לדוגמה, בתרשים 12.22 ניתן לראות פקד MonthView, שעבורו הוצב הערך 2 בשני המאפיינים, מצב אשר גורם לפקד להיות מוצג במבנה רשת.



תרשים 12.22: פקד MonthView אינו מוגבל להצגת חודש אחד ברגע נתון



אזהרה:

לא ניתן להציג יותר מ-12 חודשים במסגרת פקד MonthView ברגע נתון. אם תנסה לעשות כן ייגרם מצב שגיאה.

כדי לאפשר יותר גמישות תוכל להגדיר את המאפיינים MonthCols ו-MonthRows בזמן ריצה, כדי לאפשר למשתמשים להגדיר את מספר החודשים הרצוי להם באופן דינמי. משתמש אינו יכול לשנות את קנה המידה של תצוגת הפקד בזמן ריצה. כלומר הרשת בגודל 2X2 מוצגת תמיד בגודל קבוע. אם תשנה את גודל הרשת בזמן ריצה, גודל הפקד ישתנה כדי להתאים לגודל החדש. לכן כדאי לדעת גודל רצוי לפני הפעלת התוכנית. למרבה המזל פקד MonthView תומך בשיטה ComputeControlSize המחזירה את גודל הפקד שמכיל מספר נתון של שורות וטורים, כבדוגמה הבאה:

```
Dim sngWidth As Single, sngHeight As Single
MonthView1.computeControlSize, 4, 2, SngWidth, sngHeight
Debug.Print "New control dimensions: " & sngWidth, sngHeight
```

הערה:

המספר המוחזר על ידי השיטה ComputeControlSize נתון ביחידות המוגדרות במאפיין ScaleMode של הטופס (שהגדרת ברירת המחדל שלו היא Twips).

קטע הקוד שהוצג כלל קריאה לשיטה ComputeControlSize לשם קביעת גודל פקד MonthView בגודל 4X2. הערכים שמוחזרים באמצעות פרמטרים מוצבים במשתנים sngWidth ו-sngHeight.

עבודה עם ערכים

עד כה למדת כיצד לשלוט בתצוגת פקד MonthView. אך כדי לבצע דברים מועילים עם פקד MonthView דרושה דרך שתאפשר לזהות את התאריך ולהגדיר את התאריך העדכני. המאפיין העיקרי שבו תוכל להיעזר כדי להחזיר את התאריך העדכני וכדי להגדיר את התאריך העדכני הוא מאפיין Value, כבדוגמה הבאה:

```
MonthView1.Value = #01/03/1943#
MonthView1.Value="7/19/75 2:34:00 PM"
MonthView1.Value=Now
Debug.Print "The Selected date is " & MonthView1.Value
```

הערה:

תוכל להגדיר את טווח התאריכים החוקיים של פקד MonthView על ידי הגדרת המאפיינים MinDate ו-MaxDate של הפקד.

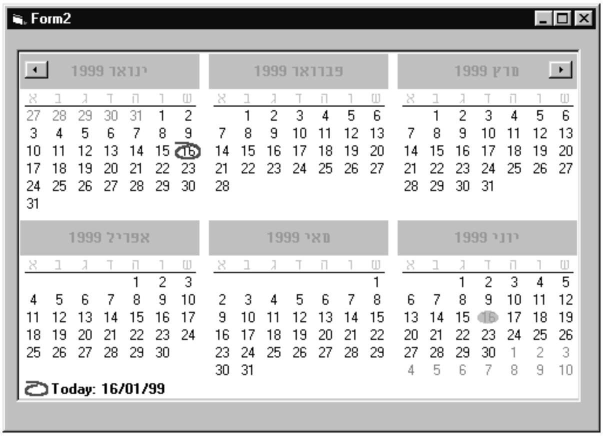
המאפיין Value תמיד מכיל ערך מסוג Date (תאריך), אך לפקד יש מספר מאפיינים נוספים אשר מייצגים רכיבים שונים של ערכו העדכני:

- Month - החודש הנבחר בשנה (המאפיין יכול להכיל ערכים שבין 1 ל -12).
- Day - היום הנבחר בחודש (המאפיין יכול להכיל ערכים שבין 1 ל -31).
- Year - השנה הנבחרת (לדוגמה 1998).
- Week - השבוע הנבחר בשנה (המאפיין יכול להכיל ערכים שבין 1 ל-52).

הערכים המאוחסנים במאפיינים שפורטו לעיל מתעדכנים באופן אוטומטי לאחר כל שינוי בערכו של מאפיין Value ולהפך. ניתן להציב במאפיינים האלה ערכים מספריים שלמים ולעבוד עם הערכים הרצויים לך.

מאפיין Value של הפקד אומנם יכול להכיל רק ערך אחד ברגע נתון, אך לפקד יש כמה מאפיינים נוספים שאותם ניתן להגדיר באופנים שיאפשרו למשתמשים לעבוד עם כמה ימים רצופים. ההצבה של ערך True במאפיין MultiSelect תאפשר למשתמשים ללחוץ על תאריך מסוים ולבחור תחום תאריכים, כמתואר בתרשים 12.23.

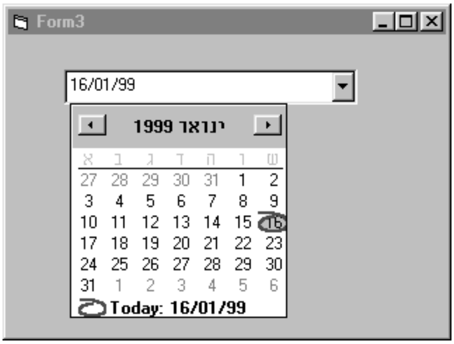
אם תגדיר את מאפיין MultiSelect לערך True תידרש להציב ערכים במאפיינים SelStart ו-SelEnd כדי להגדיר את תחום התאריכים הרצוי לך.



תרשים 12.23: מאפיין MultiSelect מאפשר לבחור תחום תאריכים

פקד DTPicker

פקד DTPicker (Date Time Picker) משלב בין תצוגת לוח השנה של פקד MonthView לבין יכולת להזנת תאריכים על ידי הקלדה. פקד זה מאפשר למשתמשים אשר יודעים את התאריך המדויק להזין אותו בקלות ובאותה עת מציע תצוגת לוח-שנה למי שזקוק לה. פקד DTPicker נראה כמו תיבה משולבת, אך כאשר לוחצים על הרשימה הנפתחת מוצג פקד MonthView.



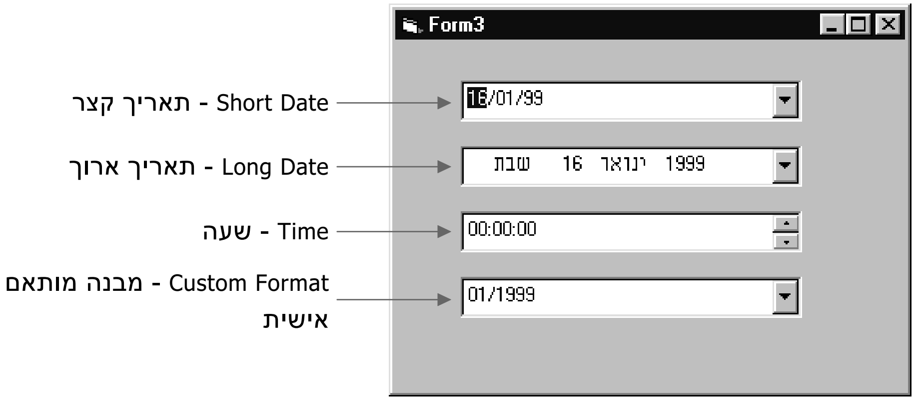
תרשים 12.24: פקד DTPicker מאפשר הזנת תאריכים באופן גרפי ובאופן טקסטואלי

הרבה ממאפייני פקד MonthView מוצעים גם במסגרת פקד DTPicker. אך בשונה מפקד MonthView, פקד DTPicker אינו יכול להציג יותר מחודש אחד ברגע נתון.

עקב העובדה שפקד DTPicker מאפשר הזנה ידנית של תאריכים, מוגדרים עבורו מספר פרמטרים המשמשים לעיצוב הקלט. תוכל להציב במאפיין Format של הפקד את הערכים המפורטים להלן:

- dtpLongDate - 0 - מציג את התאריך במבנה ארוך, בהתאם להגדרות הפעילות בלוח הבקרה.
- dtpShortDate - 1 - ברירת המחדל. מציג את התאריך במבנה קצר בהתאם להגדרות הפעילות של לוח הבקרה, לרוב M/D/YY.
- dtpTime - 2 - מציג את קטע השעה של הזמן העדכני בלבד. כאשר הגדרה זו פעילה לחצן החץ הנפתח מוחלף על ידי חיצים המצביעים למעלה ולמטה ולוח השנה אינו מוצג.
- dtpCustom - 3 - מציג את התאריך מותאם אישית.

כאשר תשתמש בהגדרה dtpCustom - 3 תוכל להציב מחרוזת עיצוב משלך במאפיין Custom. בתרשים 12.25 יוצגו כמה מבנים להצגת תאריכים.



תרשים 12.25: פקד DTPicker מאפשר להציג תאריכים תוך שימוש בכמה מבני תצוגה

גלישה למספרים

Slider Control (פקד הגררה) הוא פקד נוסף הכלול באוסף הפקדים המשותפים של Windows. פקד זה מאפשר למשתמשים להזין נתונים מספריים במסגרת תוכנית. אופן פעולת הפקד דומה לזה של מתגי החלקה שבבקרות אקולייזר של מערכות סטריאו או לחילופין לאופן הפעולה של פקד Scroll Bar (פס הגלילה) הכלול באוסף הפקדים הסטנדרטיים של Visual Basic.

צור את פקד Slider על ידי שרטוט על הטופס. לאחר מכן תידרש להגדיר ארבעה מאפיינים עיקריים שיגדירו את חזותו. מאפיינים אלה מפורטים בטבלה 12.4 ותוצאות השימוש בהם מוצגות בתרשים 12.26.

טבלה 12.4: מאפייני הפקד Slider

מאפיין	תיאור
BorderStyle	מגדיר את סוג הגבול שיוצב. הצבת 0 במאפיין גורמת לכך שלא יוצג גבול והצבת ערך 1 בפקד גורמת לכך שיוצג קו גבול יחיד
Orientation	מגדיר את פריסת הפקד. הגדרת המאפיין כ-0 מציגה מחוון אופקי והצבת ערך 1 במאפיין מציגה מחוון אנכי
TickStyle	מגדיר את מיקום Tick Marks (שנתות) על גבי המחוון. הצב 0 במאפיין להצגת שנתות מתחת למחוון או מימינו, 1 להצגת שנתות מעל המחוון או משמאלו, 2 להצגת שנתות משני צידי המחוון ו-3 אם אינך מעוניין להציג שנתות
TickFrequency	קובע את מספר השנתות שתוצגנה. ניתן להציב כל מספר חיובי

לאחר שתגדיר את חזות הפקד תידרש להגדיר מספר מאפיינים שיגדירו את אופן פעולתו. Min ו-Max הם מאפיינים ראשונים. מאפיינים אלה מגדירים את טווח הערכים שהפקד יוכל לקבל. שני המאפיינים הבאים הם LargeChange ו-SmallChange. מאפיין LargeChange מגדיר את שיעור השינוי בערך הפקד שייגרם כתוצאה מהקשה על מקש PageUp או PageDown או לחיצה על לחצן העכבר כשמצביע העכבר מצוי באחד מצידי המחוון. מאפיין SmallChange מגדיר את שיעור השינוי בערך הפקד שייגרם עקב הקשה על אחד ממקשי החיצים. מאפיין SelectRange הוא המאפיין האחרון המשמש להגדרת אופן פעולת הפקד. מאפיין זה קובע האם הפקד יוכל לשמש לשם בחירת ערך יחיד או לשם בחירת טווח ערכים.



תרשים 12.26: פקד Slider מאפשר להזין נתונים מספריים באופן גרפי

שימוש בפקד Slider

משתמשים יוכלו להגדיר ערך לפקד Slider באמצעות לחיצה עליו או גרירתו. הם יוכלו גם להיעזר במקשים PageUp, PageDown, חץ ימינה או חץ שמאלה לשינוי ערך הפקד. הנתון שיוזן על ידי המשתמש יישמר במאפיין Value של הפקד.

הערה:

מקשים משפיעים על פקד Slider רק אם הוא מצוי במוקד (Focus).

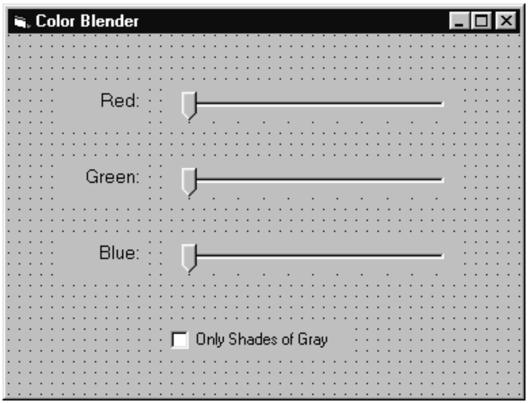
אם מאפיין SelectRange מוגדר True יתאפשר למשתמשים לבחור תחום ערכים על ידי החזקת מקש Shift לחוץ בעת גרירת המחוון. אך לרוע המזל זוהי אינה פעולה שמתבצעת באופן אוטומטי, אלא פעולה שיש לכתוב קוד שיבצע אותה. תידרש להוסיף לשגרת אירוע MouseDown של הפקד קוד שיזהה האם מקש Shift לחוץ בעת גרירת מחוון הפקד. אם כן, הקוד יגדיר את המאפיין SelStart של הפקד שמכיל את נקודת ההתחלה של התחום הנבחר ואת מאפיין SelRange שמגדיר את גודל התחום. בעזרה המקוונת של Visual Basic תוכל למצוא דוגמה טובה של שימוש במאפיין זה.

פרויקט Color Blender

נדגים את השימוש בפקד Slider על ידי בניית פרויקט הדוגמה Color Blender (מערבל צבעים). ידוע לך בוודאי שלטפסים של Visual Basic יש מאפיין BackColor שמכיל ערך מספרי אשר מייצג את צבע הרקע של הטופס. פרויקט הדוגמה הפשוט הזה ישתמש בשלושה פקדי Slider להגדרת מאפיין BackColor של הטופס.

צור את הפרויקט על ידי ביצוע הפעולות הבאות:

1. צור פרויקט חדש מסוג Standard EXE.
2. הוסף לטופס פקד **CheckBox**. הגדר את מאפיין **Name** של הפקד **ckGray** ואת מאפיין **Caption** שלו **OnlyShades of Gray**.
3. הוסף לטופס פקד **Slider** והגדר את שמו כ-**sldColor**.
4. שרטט פקד **Label** משמאל לפקד Slider והגדר את שמו **lblColor**.
5. צור שני ***מערכי פקדים*** (Control Arrays) לפקדים lblColor ו-sldColor. צור את מערכי הפקדים על ידי בחירת שני הפקדים כאשר מקש Ctrl לחוץ. בחר **Copy** מתפריט **Edit** ולאחר מכן **Paste**. תוצג לפניך תיבת דו-שיח אשר תשאל אותך האם אתה מעוניין ליצור מערך פקדים. לחץ על **Yes** (מערכי פקדים יתוארו בפירוט בפרק 13 **עבודה עם מערכי פקדים**).
6. הוסף פריט שלישי לכל אחד ממערכי הפקדים על ידי כך שתבחר שוב **Paste**.
7. הגדר את מאפיין **Caption** של **lblColor(0)** כ-**Red**, את מאפיין **Caption** של **lblColor(1)** כ-**Green** ואת מאפיין **Caption** של **lblColor(2)** כ-**Blue**.
8. סדר את מערכי הפקדים **(lblColor** ו-**sldColor** במבנה המוצג בתרשים 12.27.



תרשים 12.27: הטופס הראשי של פרויקט Color Blender מכיל מערך של פקדי Slider

לאחר שתיצור את ממשק התוכנית, תידרש להוסיף לתוכנית כמה אירועים שיאפשרו לה לבצע את פעולתה. האירוע הראשון הוא אירוע Load של הטופס, שהקוד שלו מובא בתוכנית 12.5. האירוע הזה יוצר את המצב ההתחלתי של התוכנית.

תוכנית 12.5: SLIDERS.VBP - אתחול התוכנית במסגרת אירוע Load של הטופס

```
Private Sub Form_Load()
      Dim nCount As Integer
      Dim lRed As Long, lGreen As Long, lBlue As Long
'Set up the sliders
      For nCount = 0 To 2
            sldColor(nCount).Min = 0
            sldColor(nCount).Max = 255

            sldColor(nCount).TickFrequency = 4
      Next nCount

      'Initialize the color of the form
      'control. (This will set the color to black. RGB(0,0,0))
      Me.BackColor = RGB(lRed, lGreen, lBlue)

End Sub
```

לאחר מכן תוסיף את הקוד שיבצע את פעולת שינוי הצבע כתגובה להזזת המחוון של הפקד. הוסף את הקוד המובא בתוכנית 12.6 בשגרת האירוע Scroll של מערך Slider Controls.

```
Private Sub sldColor_Scroll(Index As Integer)
	Dim nCount As Integer
	Dim lRed As Long, lGreen As Long, lBlue As Long

	'Gray is equal values of Red, Green and Blue
	If ckGray Then
		'move all the sliders
		For nCount = 0 To 2
			sldColor(nCount).Value = sldColor(Index).Value
		Next nCount
	End If

	'Set the RGB value
	lRed = sldColor(0).Value
	lGreen = sldColor(1).Value
	lBlue = sldColor(2).Value

	'Assign the resultant RGB value to the backcolor
	Me.BackColor = RGB(lRed, lGreen, lBlue)

End Sub
```

הפעלת הפרויקט

לאחר שתזין את קוד הפרויקט הקש על F5 כדי להדר את התוכנית ולהפעיל אותה. נסה להציב שילובים שונים של ערכים בפקדי Slider כדי לראות את השפעתם על צבע הרקע שיווצר.

אופן הפעולה של פרויקט Color Blender הוא כדלקמן: לאחר שתהליך האתחול של הטופס מתבצע במסגרת האירוע Form_Load, רוב העבודה המעשית של היישום מתבצעת במסגרת אירוע Scroll של מערך הפקדים sldColor. לאחר שהמשתמש מזיז מחוון פקד הכלול במערך הפקדים, אינדקס הפקד שהוזז מועבר אל שגרת האירוע sldColor_Scroll. היישום בודק את מאפיין Value של תיבת הסימון ckGray. אם תיבת הסימון מסומנת (כלומר Value = vbChecked), ערך הפקד Slider שהוזז מוצב בכל פקדי Slider הכלולים במערך הפקדים. שינוי ערך זה גורם לכל מחווני הפקדים לנוע אל אותו ערך, לפני שקוד אחר כלשהו מתבצע.

לאחר מכן, הערך המאוחסן במאפיין Value של כל אחד מפקדי Slider הכלולים במערך הפקדים מוצב במשתנה הצבע המתאים - lRed, lGreen ו-lBlue. המשתנים שמייצגים את הצבעים מועברים כפרמטרים לפונקציה ()RGB והערך המוחזר מהפונקציה מוצב במאפיין BackColor של הטופס.

דיווח על מצב והתקדמות התוכנית

אחד התחומים החשובים בפעולת התוכנית הוא הצגת המתרחש בכל זמן נתון. בעת שימוש בתוכנית כגון מעבד תמלילים, המשתמשים מעוניינים לדעת באיזה עמוד הם מצויים כעת ומהו מיקומם במסגרת העמוד הזה. נתונים כאלה המכונים **נתוני מצב** (Status Information) מוצגים לרוב באופן רציף על המסך ולא במסגרת תיבת הודעה. המשתמשים גם מעוניינים לדעת כמה זמן תימשך פעולה מסוימת ואיזה מרכיב של הפעולה מתבצע באותו רגע נתון. נתונים כאלה מכונים **נתוני התקדמות** (Progress Information). בסעיף זה נסקור שלושה פקדים משותפים המשמשים לצרכים אלה:

- StatusBar, שמציג הודעות אודות מצב התוכנית.
- ProgressBar, אשר מציג חיווי גרפי של שיעור ההתקדמות לקראת מטרה מסוימת.
- Animation, שמאפשר להציג "סרטונים" שיהוו אינדיקציה לכך שהתוכנית מבצעת פעולה כלשהי.

לפני שיתאפשר לך להשתמש בפקדים אלה תידרש לצרף אותם לארגז הכלים של Visual Basic. תיאור הפעולות הדרושות לשם הוספת הפקדים לארגז הכלים נמצא בסעיף **מבוא לפקדים משותפים** בתחילת הפרק.

הוספת שורת מצב לתוכנית

בימים שקדמו להמצאת פקד StatusBar (שורת מצב), מפתחי Visual Basic נהגו להשתמש בשורות מצב "מזויפות" שהיו מורכבות מ"לוחיות" (Panels) ופקדי Label. פקדים כאלה פעלו די טוב (וישנם אף כאלה שפועלים גם כיום), אך פקד StatusBar הוא תחליף נוח יותר לשימוש עבור פקדים כאלה. אם כבר השתמשת במסגרת היישום שלך בפקד משותף כלשהו מהקבוצה בה כלול פקד StatusBar, לא תידרש להתייחס לתקורה (Overhead) העלולה לנבוע מהצורך בהפצת פקד נוסף מפני שכל הפקדים של הקבוצה מאוחסנים בקובץ OCX אחד, שאותו תידרש להפיץ בלאו הכי.

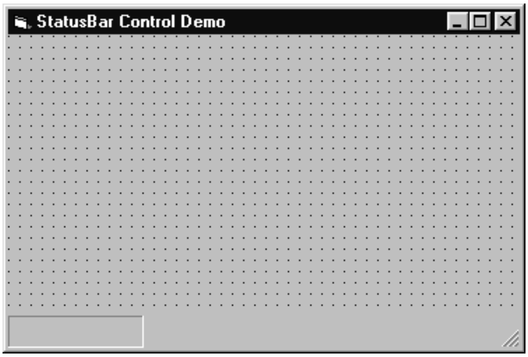
תרשים 12.28 מציג פקד StatusBar טיפוסי המשמש במסגרת יישום הכתוב בשפת Visual Basic.



תרשים 12.28: שורות מצב מוצגות לרוב בתחתית הטופס

יצירת שורת מצב

כדי ליצור שורת מצב תידרש תחילה לבחור את פקד StatusBar ואחר כך לשרטט אותו על גבי הטופס שלך. בתחילה פקד StatusBar משתרע על פני כל רוחב הטופס שאליו הוא מוסף והוא מכיל רק לוחית אחת, כפי שניתן לראות בתרשים 12.29. בוודאי שמת לב שלא חשוב היכן תשרטט את פקד StatusBar, הוא תמיד יוצג בתחתית הטופס. תחתית הטופס היא המיקום המקובל של שורות מצב.



תרשים 12.29: התצורה של שורת מצב מוגדרת באופן אוטומטי בעת הוספתה לטופס

לאחר שתשרטט את פקד StatusBar, תוכל להתחיל בהגדרת המאפיינים שיקבעו את חזותו, וביצירת הלוחיות הנוספות הדרושות לשם הצגת הנתונים. ראית כבר שפקד StatusBar נמתח על פני רוחב הטופס, כאשר הוא משורטט. אך את גובה הפקד תוכל להגדיר על ידי הצבת ערך במאפיין Height שלו או על ידי בחירתו וגרירה של אחת מידיות הגדרת הגודל.

קיימים שני מאפיינים נוספים אשר מגדירים את החזות הבסיסית של פקד StatusBar: Style ו-Align.

המאפיין Align מגדיר לאיזה קצה של הטופס תעוגן שורת המצב (אם היא בכלל תעוגן). המאפיין יכול לקבל חמישה ערכים אפשריים:

- vbAlignNone - מאפשר למקם את שורת המצב בכל מקום על גבי הטופס.
- vbAlignTop - ממקם את שורת המצב בקצה העליון של הטופס.
- vbAlignBottom - ממקם את שורת המצב בקצה התחתון של הטופס. זהו ערך ברירת המחדל של המאפיין, אשר מייצג את המקום בו המשתמשים מצפים למצוא את שורת המצב.
- vbAlignLeft - ממקם את שורת המצב בקצה השמאלי של הטופס.
- vbAlignRight - ממקם את שורת המצב בקצה הימני של הטופס.

הערה:

אני ממליץ למקם את שורת המצב בקצה התחתון של הטופס, בכל מקרה שבו לא קיימת סיבה המונעת לעשות כן. כל מיקום אחר עשוי אומנם להוות חידוש מקורי אך הוא גם עלול לבלבל את המשתמשים, שרגילים למצוא את שורת המצב בקצה התחתון של מסך היישום.

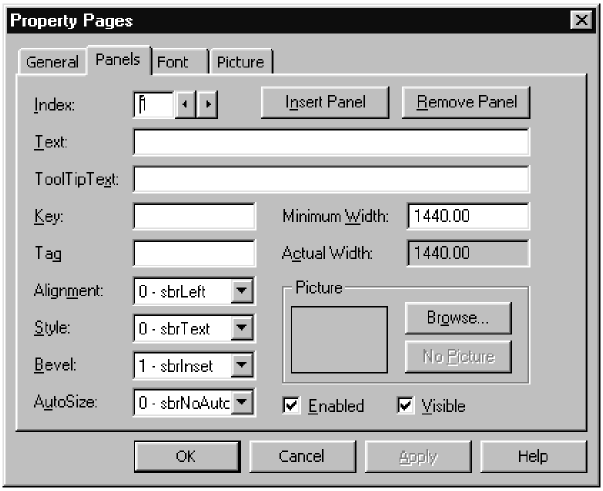
בתרשים 12.28 ראית שניתן להציג בשורת המצב מספר הודעות בבת-אחת. כל מקטע של שורת המצב מכונה **לוחית** (Panel). להלן שני ערכים שאותם ניתן להציב במאפיין Style של הפקד כדי להגדיר האם הפקד יכיל לוחית אחת או כמה לוחיות:

- sbrNormal - מאפשר הצגת מספר לוחיות במקביל.
- sbrSimple - מאפשר הצגת לוחית אחת בלבד.

אם תגדיר את מאפיין Style כ-sbrSimple, פקד StatusBar יתפקד כמו פקד Label, שהכיתוב שיוצג במסגרתו ייקבע על ידי הגדרת מאפיין SimpleText של הפקד. אך בעת שימוש בשורות מצב שתכלנה מספר לוחיות תוכל להשתמש ביכולות נוספות.

עבודה עם לוחיות של שורת מצב

הפעולה הממשית של שורת מצב מבוצעת על ידי הלוחיות הכלולות בה. כל לוחית היא אובייקט נפרד, עבורו מוגדרים מאפיינים ואופני פעולה משלו. כל לוחית היא פריט באוסף Panels של פקד StatusBar. לאחר שתשרטט את פקד StatusBar ותגדיר את המאפיינים שלו, תוכל להתחיל להוסיף לו לוחיות. הוספת הלוחיות תתבצע באמצעות גיליון המאפיינים של פקד StatusBar אשר מוצג בתרשים 12.30.



תרשים 12.30: תוכל לנהל את שורת המצב על ידי לחיצה על הלחצן בונה (עם שלוש נקודות) המופיע בימין שורת המאפיין Custom שבגיליון המאפיינים

תוכל להיעזר בתיבת הדו-שיח Property Pages של הפקד, להגדרת המאפיינים של כל לוחית בנפרד. שמונת המאפיינים המפורטים להלן, הם המאפיינים העיקריים המשמשים להגדרת החזות ואופן פעולת הלוחית:

- Text - מגדיר את הטקסט שיוצג על גבי לוחית שמאפיין Style שלה יוגדר כ-sbrText (מקובל להגדיר את מאפיין Text בזמן ריצה).
- ToolTipText - מגדיר את הטקסט שיוצג כאשר המשתמש ישהה את מצביע העכבר על גבי הלוחית.
- Alignment - מגדיר את אופן היישור של הטקסט שיוצג במסגרת הלוחית (לשמאל, לימין או ממורכז).
- Style - מגדיר את סוג הלוחית שיווצר.
- Bevel - מגדיר את אופן ההצללה שייושם בעת תצוגה תלת-מימדית של הלוחית.
- AutoSize - מגדיר את האופן בו התוכנית תקבע את גודל הלוחית.
- MinWidth - קובע את הגודל המינימלי של הלוחית.
- Picture - מגדיר את התמונה שתוצג במסגרת הלוחית, אם קיימת כזו.

רוב המאפיינים האלה הם קלים להבנה ואינם דורשים הסבר נוסף, אך כדאי שנביא מעט פירוט נוסף אודות המאפיינים Style וכן AutoSize.

ניתן להגדיר שבעה סוגים שונים של לוחיות במסגרת שורת מצב. אחד הסוגים (sbrText) מיועד להצגת טקסט שיוגדר על-ידך, אך שאר הסוגים מיועדים להצגת פריטי מידע מוגדרים מראש. סוגים אלה משמשים להצגת מידע אודות מצב מקשי הנעילה (Caps Lock, Num Lock, וכדומה) ולהצגת התאריך והשעה העדכניים. לוחיות מסוגים אלה מתופעלות על ידי הפקד עצמו, כך שאתה תידרש בסך הכל להגדיר אותן בעת עיצוב היישום והשימוש בהם לא ידרוש ממך לבצע פעולה כלשהי בעת פעולת התוכנית. סוגים אלה מפורטים בטבלה 12.5 ואילו בתרשים 12.28 תוכל לראות דוגמה של כל אחד מהם.

הערה:

מאפיין שמיני, sbrKana יכול לקבל ערך נוסף שיוצג בלוחית מקש Scroll Lock. הצגת הכיתוב KANA בכתב מודגש תציין שהמקלדת פעילה במצב הוספת אותיות יפניות.

מאפיין AutoSize של לוחית יכול לקבל אחד משלושה ערכים, אשר יקבעו את גודל הלוחית. הגדרת ברירת המחדל, NoAutoSize מציגה את הלוחית בגודל המוגדר במסגרת מאפיין MinWidth שלה. גודל זה לא ישתנה בעקבות הוספה/גריעה של לוחיות לשורת המצב. הצבת ערך sbrContents במאפיין תגרום לכך שרוחב הלוחית יותאם לרוחב הטקסט שיוצג במסגרתה, אם זה טקסט שהוגדר על ידי התוכנית, או אם מדובר במחווני מצב המקשים או במחווני התאריך/שעה. הערך האחרון, sbrSpring מאפשר להרחיב את הלוחיות כדי להתאים את רוחבן לרוחב שורת המצב. שימוש בערך הזה מונע משורת המצב להכיל שטח פנוי כלשהו.

טבלה 12.5: סוגי לוחיות הזמינים לשימוש במסגרת תוכניות

הגדרה	תיאור סוג
sbrText	מציג טקסט או תמונת Bitmap. הטקסט שיוצג במסגרת הלוחית מאוחסן במאפיין Style שלה ואילו התמונה מאוחסנת במאפיין Picture שלה
sbrCaps	מציג את מצב מקש Caps Lock. אם המקש מופעל מוצג על הלוחית הכיתוב CAPS מודגש ואילו כאשר המקש מנוטרל הכיתוב מוצג מעומעם
sbrNum	מציג את מצב מקש Num Lock. אם המקש מופעל מוצג על הלוחית הכיתוב NUM מודגש ואילו כאשר המקש מנוטרל הכיתוב מוצג מעומעם
sbrIns	מציג את מצב מקש Insert. אם המקש מופעל מוצג על הלוחית הכיתוב Ins מודגש ואילו כאשר המקש מנוטרל הכיתוב מוצג מעומעם
sbrScrl	מציג את מצב מקש Scroll Lock. אם המקש מופעל מוצג על הלוחית הכיתוב SCRL מודגש ואילו כאשר המקש מנוטרל הכיתוב מוצג מעומעם
sbrTime	משמש להצגת השעה העדכנית
sbrDate	מציג את התאריך העדכני

הערה:

ללא תלות בערך אותו תציב במאפיין AutoSize לא יתאפשר לך להגדיר לוחית שרוחבה יהיה קטן מהערך המוצב במאפיין MinSize.

המאפיין האחרון שאליו נתייחס הוא מאפיין Bevel. כבר ציינו שזהו המאפיין המשמש להגדרת האופן בו יוצגו האפקטים החזותיים התלת-מימדים במסגרת הלוחית. המאפיין הזה יכול לקבל שלושה ערכים: sbrNoBevel אשר מציג לוחית שטוחה, sbrInset (ערך ברירת המחדל של המאפיין) אשר מציג לוחית שנראית שקועה בשורת המצב ו-sbrRaised אשר מציג לוחית מובלטת. שלושת הסגנונות מוצגים בתרשים 12.31.



תרשים 12.31: תוכל לכלול בשורת המצב לוחיות שטוחות, לוחיות שקועות או לוחיות מובלטות

שליטה על לוחיות על ידי שימוש בקוד

למדת איך להגדיר מאפיינים של כל לוחית בנפרד. אך עליך עדיין ללמוד כיצד להוסיף ולגרוע לוחיות לשורת המצב. בעת עבודה עם גיליון המאפיינים של פקד StatusBar תוכל להיעזר בלחצנים Insert Panel ו- Remove Panel. אך ניתן גם להוסיף ולגרוע לוחיות על ידי שימוש בקוד. פעולות אלו תבוצענה על ידי שימוש בשלוש שיטות האוסף Panels (Panels Collection) :

- Add - יוצרת לוחית חדשה שתיכלל בשורת המצב.
- Delete - גורעת לוחית נתונה משורת המצב.
- Clear - גורעת את כל הלוחיות הכלולות בשורת המצב.

הערה:

בשונה משאר האוספים (Collections) של Visual Basic, ערך מאפיין Index של הפריט הראשון באוסף Panels הוא אחד וערך מאפיין Index של הפריט האחרון שווה לערך המאוחסן במאפיין Count של האוסף.

ראית כבר שהלוחיות המשמשות להצגת נתוני מצב המקשים ולהצגת נתוני התאריך והשעה מבצעות פעולות אלו באופן אוטומטי, ואלו אכן פעולות חשובות. אך כוחה האמיתי של שורת המצב נעוץ ביכולת שיש לה לשנות את הטקסט המוצג במסגרת לוחיות מסוג Text, תוך כדי פעולת התוכנית. לוחיות אלו מורות למשתמש מהו מצב התוכנית שעמה הוא עובד.

אם תרצה לעדכן תצוגת לוחית תוך שימוש בקוד תוכנית, תוכל להציב מחרוזת במאפיין Text של אותה לוחית, כפי שמתואר בשורת הקוד הבאה :

```
StatusBar1.Panels(1).Text = "Viewing record 1 of 10"
```

תוכל כמובן להגדיר גם מאפיינים נוספים באותו אופן. תוכל אפילו להגדיר את שורת המצב כולה על ידי שימוש בקוד, תוך שימוש בשיטות של אוסף Panels ובמאפיינים של אובייקטי Panel. הקוד המובא בתוכנית 12.7 ייצור את שורת המצב המוצגת בתרשים 12.32. קטע הקוד הזה כולל שימוש במאפיין Count של האוסף Panels, שבו מאוחסן מספר הלוחיות הכלולות בשורת המצב.

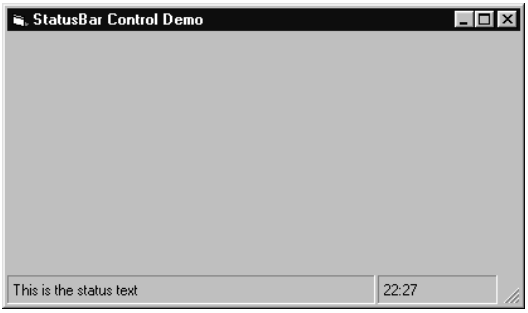
תוכנית 12.7: STATUS.VBP - הגדרת שורת המצב בעזרת קוד

```
Private Sub Form_Load()
    With StatusBar1
        'Remove the default panel
        .Panels.Remove 1
        'Add a Text panel at the left
        .Panels.Add 1, "mykey", "This is the status text", sbrText
        .Panels(1).AutoSize = sbrSpring
```

```
            'Add the current time display
            .Panels.Add,,, sbrTime

    End With

End Sub
```



תרשים 12.32: תוכל לשלוט על שורת המצב על ידי הגדרה של מאפייני לוחיות

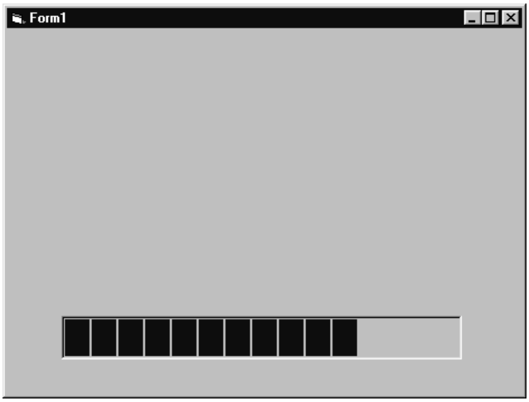
שורת התקדמות

פקד StatusBar אינו הפקד היחיד שבו תוכל להשתמש לצורך מעקב אחר פעולת התוכנית. יש תוכניות אשר נדרשות לבצע פעולות שאורכות זמן רב, כך שתרצה שלתוכנית תהיה יכולת לספק למשתמשים חיווי דינמי של מצב התקדמות הפעולה. פקד **ProgressBar** (מד התקדמות) הוא כלי אידיאלי לשימוש במצבים כאלה. לדוגמה, Internet Explorer נעזר בפקד ProgressBar לשם הצגת אחוז מגודל הקבצים שהועברו עד כה. במצב כזה פקד ProgressBar מאפשר לך לדעת שני דברים: ראשית, ההשתנות של התצוגה בפקד מהווה עבורך אינדיקציה שביצוע הפעולה נמשך. שנית, תצוגת הפקד תאפשר לך להסיק עוד כמה זמן יידרש להשלמת הפעולה.

הגדרת פקד ProgressBar

קל מאוד להגדיר פקד ProgressBar ולהשתמש בו. בדומה לפקדים אחרים תידרש תחילה לשרטט את הפקד על גבי הטופס ולאחר מכן תידרש להגדיר את מאפייני הפקד. Height ו-Width הם שני המאפיינים החשובים מקרב המגדירים את חזות הפקד. רוחב מד התקדמות הוא ברוב המקרים גדול מאוד ביחס לגובה. למעשה, בעזרה המקוונת של פקד ProgressBar כלולה המלצה שרוחב מד ההתקדמות יהיה גדול לפחות פי 12 מגובהו. בתרשים 12.33 מוצג מד התקדמות טיפוסי.

מאפיין Value של פקד ProgressBar מכיל ערך אשר מגדיר את השיעור מתוך הפקד שיוצג כשהוא מלא. אם נדמה פקד ProgressBar למדחום אזי ניתן להתייחס אל הערך המאוחסן במאפיין Value כאל הטמפרטורה הנוכחית. השטח שמתחת לטמפרטורה זו מלא בכספית ואילו שאר השטח ריק.



תרשים 12.33: מדי התקדמות מספקים חיווי חזותי לשיעור הפעולה שכבר הושלם

Max ו-Min הם המאפיינים הנוספים שאותם תידרש להגדיר עבור פקד ProgressBar. הערכים שיאוחסנו במאפיינים אלה ייצגו את הערך המקסימלי והמינימלי שאותם ניתן יהיה לאחסן במאפיין Value של הפקד. אם נחזור לרגע לדימוי מד החום אזי הערכים המאוחסנים במאפיינים אלה מייצגים את הטמפרטורה המקסימלית והמינימלית שאותו מדחום יוכל למדוד. לדוגמה, הצבת ערך 50 במאפיין Value כאשר המאפיין Min מוגדר כ-0 והמאפיין Max מוגדר כ-100 תגרום להצגת מד ההתקדמות כשהוא מלא עד מחציתו.

במאפיינים Max ו-Min ניתן להציב כל ערך שלם חוקי. אך הערך שיוצב במאפיין Max חייב להיות גדול מזה שיוצב במאפיין Min. תוכל להציב בשני המאפיינים האלה כל ערך שיהיה בעל משמעות במסגרת היישום שלך. להלן שתי דוגמאות לשימוש במאפיינים אלה:

- בעת עיבוד רשומות הכלולות במסד נתונים כדאי להציב 0 במאפיין Min ולהציב במאפיין Max את מספר הרשומות שאותן נידרש לעבד. הגדלת הערך המאוחסן במאפיין Value לאחר סיום עיבוד כל רשומה יאפשר להציג לפני המשתמשים חיווי של התקדמות הפעולה.
- בעת ביצוע פעולה להורדת קובץ תוכל להציב במאפיין Max את גודל הקובץ בקילו בתים או את מספר הבלוקים שהקובץ יתפוס. אופן שימוש אחר במאפיינים אלה במסגרת פעולות להורדת קבצים הוא על ידי הצבת 0 במאפיין Min והצבת ערך 100 במאפיין Max והתייחסות לאחוזים מתוך הקובץ שהורדו עד כה.

עדכון מד ההתקדמות בעת פעולת התוכנית

הגדרת מאפיין Value היא המפתח להצגת שיעור ההתקדמות של פעולה המבוצעת כעת. כאשר קוד התוכנית שלך יבצע פעולה מסוימת תידרש לעדכן באופן סדיר את הגדרת המאפיין Value. כפי שציפית, תוכל לבצע את פעולת העדכון במסגרת לולאה שתבצע פעולות שתחזורנה על עצמן, כפי שקורה בדוגמת הקוד הבאה:

```
'In the following code "rsMain" is a recordset
'that has already been opened and populated.
'"pbr" is the ProgressBar control.

pbr.Min=0
rsMain.MoveLast
pbr.Max = rsMain.RecordCount
rsMain.MoveFirst
inCounter = 0

Do While Not rsMain.EOF
'Insert code to process Records here
	inCounter = inCounter +1
	pbr.Value = inCounter
	rsMain.MoveNext
Loop
```

שילוב קטעי וידאו באמצעות פקד Animation

פקד **Animation** (הנפשה) מאפשר לך לכלול בתוכניות יכולות הנפשה. פקד זה יכול להציג קבצי AVI אילמים. קובץ AVI מכיל למעשה סדרת קבצי תמונה אשר מוצגים ברצף, באופן היוצר אפקט הנפשה, באופן דומה לזה שמיושם בסרטים מצויירים. במקרים מסוימים תוכל להיעזר בפקד Animation לצורך הצגת חיווי שיאפשר למשתמש לדעת שמתבצעת כעת פעולה כלשהי, כפי שקורה במסגרת פעולת העתקת קבצים בסביבת Windows (ראה תרשים 12.34). קטעי ההנפשה יוצגו ברקע של סביבת העבודה, במקביל להתרחשות פעולות אחרות.



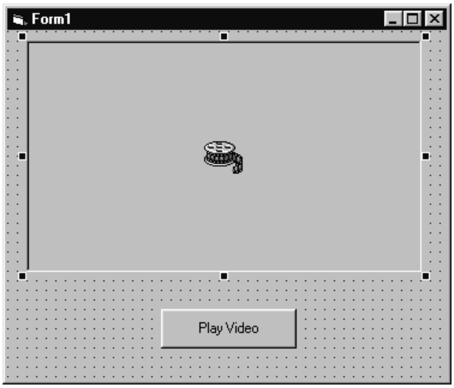
תרשים 12.34: התקדמות פעולת העתקה מוצגת על ידי שימוש בקטע הנפשה פשוט

הערה:

ניתן ליצור קטעי הנפשה פשוטים גם על ידי שימוש בפקד Timer לשינוי מיקום פקד תמונה מדי פרק זמן קבוע. ליישום שיטה זו תידרש אומנם לכתוב מעט יותר קוד מכפי שתידרש לכתוב לצורך שימוש בפקד Animation, אך השימוש בה יחסוך את הצורך להכין את קטע ההנפשה כקובץ AVI. פקד Timer תואר בפרק 4, **שימוש בפקדי ברירת המחדל של Visual Basic**.

הגדרת פקד Animation

כדי להגדיר את פקד Animation תידרש תחילה לשרטט מופע של הפקד על גבי הטופס שלך. בתרשים 12.35 נראית התצוגה ההתחלתית של פקד Animation.



תרשים 12.35: התצוגה ההתחלתית של פקד Animation דומה לפקד תיבת תמונה שבמרכזו מוצג גלגל סרט

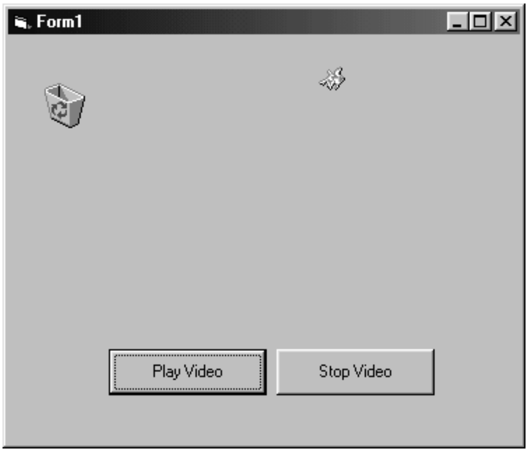
שרטוט הפקד על גבי הטופס מגדיר אובייקט שיכיל את קטע האנימציה. אך כדי להפעיל את קטע ההנפשה תידרש לפתוח קובץ הנפשה ולהריץ אותו.

את הפעולות האלו תוכל לבצע על ידי הוספת לחצן פקודה הכללת קוד בשגרת אירוע Click שלו אשר יפתח את קובץ הווידאו ויציג אותו.

לפתיחת הקובץ תוכל להשתמש בשיטה Open של פקד Animation, שבקריאה אליה תכלול הגדרה של נתיב הגישה אל הקובץ שאותו תרצה להציג (תוכל למצוא כמה קבצי וידאו בתיקיה Graphics\Videos\ של Visual Basic, בהנחה שהתקנת אותם). לאחר שתפתח את קובץ AVI, הפעל את השיטה Play של הפקד כדי להתחיל בהצגת קטע ההנפשה. שתי שורות הקוד המוצגות כאן פותחות קובץ AVI ומציגות אותו:

```
Animation1.Open "C:\Progra~1\Micros~1\Common\Graphics\Videos\FileNuke.Avi"
Animation1.Play
```

בתרשים 12.36 מוצג פקד Animation בעת הצגת קטע וידאו. אם תרצה לעצור את הצגת קטע ההנפשה תוכל לקרוא לשיטה Stop של הפקד.



תרשים 12.36: קריאה לשיטה Play תגרום להצגה חוזרת ונשנית של קטע הווידאו

טיפ:

כדאי לך אולי ליצור טופס שיכלול כלים להצגת נתוני מצב שבו תוכל להשתמש במסגרת כל הפרויקטים שתכתוב ב- Visual Basic. בקובץ כזה תוכל לכלול פקד ProgressBar ופקד Animation. תוכל להגדיר את מאפיין AutoPlay של פקד Animation לערך True, כדי להציג את קובץ הווידאו באופן אוטומטי, מבלי שתידרש לפנות לשיטה Play.

פרמטרים אופציונליים של השיטה Play

אופן הפעולה שהוא ברירת מחדל של פקד Animation הוא הצגת קטע וידאו בלולאה, כלומר הקטע מוצג שוב ושוב, ללא הפסקה עד שמתבצעת קריאה לשיטה Stop. להלן שלושה פרמטרים אופציונליים של השיטה שיאפשרו לשנות אופן פעולה זה:

- Repeat - מגדיר את מספר הפעמים שקובץ הווידאו יוצג.
- Start - מגדיר את התמונה שבה תחל הצגת הקובץ.
- Stop - מגדיר את התמונה שבה תיעצר הצגת הקטע. תוכל לברר את מספרי התמונות בקטע על ידי שימוש בנגן המדיה של Windows (MSPLAYER.EXE).

הפרמטרים האופציונליים שפורטו כאן ישמשו במסגרת שורת הקוד שתוצג להלן, אשר תציג פעמים את תמונות 5 עד 15 של קטע ההנפשה:

```
anmAviPlayer.Play 2, 5, 15
```

תוכל לכלול בשורת הקריאה לשיטה ערכים עבור כל הפרמטרים האופציונליים האלה או עבור חלקם. בפרמטרים שלא ייכללו עבורם ערכים בשורת הקריאה לשיטה, יוצבו ערכי ברירת המחדל שלהם. סדר הפירוט של הפרמטרים הוא Repeat, Start, Stop. אם לא תרצה להגדיר ערך עבור אחד הפרמטרים האלה יהיה עליך להשתמש בפסיק כבמציין מקום.

בתקליטור:

הדוגמה הרלוונטית לפקד Animation נמצאת בתקליטור בתיקיה AnimPlayer שבפרק 12. כמו כן יש בדוגמה קטע וידאו בשם FileNuke.avi אשר צריך להפנות אליו את הפקד או לחילופין להעביר אותו אל כונן C לתיקיית השורש.

מכאן...

פרק זה הציג בפניך את קבוצת הפקדים המכונים Windows Common Controls. כמה מבין הפקדים שנסקרו בפרק זה נכללו לראשונה בשפת Visual Basic 6.0. תוכל להשתמש בפקדים האלה לשם העשרת הפרויקטים שלך במיגוון אופנים. הדרך הטובה ביותר ללמוד אודות הפקדים האלה היא על ידי התנסות בשימוש בהם במסגרת תוכניות. מידע נוסף אודות נושאים רלוונטיים תוכל למצוא בפרקים הבאים:

- אם טרם למדת אודות הפקדים המוצעים בארגז הכלים של Visual Basic כברירת מחדל, כדאי שתעיין בפרק 4 **שימוש בפקדי ברירת המחדל של Visual Basic**.
- אם תרצה ללמוד אודות פקד ToolBar (שלא תואר בפרק זה) ואודות היצירה של תפריטים, עיין בפרק 6 **שליטה נוספת למשתמש: תפריטים וסרגלי כלים**.
- מידע אודות שימוש בפקדים במסגרת מערך יובא בפרק 13 **עבודה עם מערכי פקדים**.

13

עבודה עם מערכי פקדים

מה בפרק?

- מבוא למערכי פקדים
- יצירת מערך פקדים
- עבודה עם מערכי פקדים
- יצירת מערך פריטי תפריטים
- טעינה והסרה של פקדים בזמן פעולת התוכנית

עד עתה למדת איך להשתמש בפקדים כדי לאפשר לתוכניותיך ליצור קשר עם המשתמש. הפקדים שהוספת לטפסים עד עכשיו היו עצמאיים לגמרי, ללא קשר מיוחד לפקדים אחרים בטפסי התוכנית. כאשר תיצור ממשקי משתמש מורכבים יותר, יש להניח שיהיה לך נוח יותר לעבוד עם פקדים המאוחדים לקבוצות.

בפרק 4 **שימוש בפקדי ברירת המחדל של Visual Basic**, למדת כיצד לבחור מספר פקדים בבת-אחת בסביבת העיצוב ולשנות את מאפייניהם כקבוצה אחת. למרות ששיטה זו נוחה בזמן עיצוב התוכנית, אין בכך כדי לאפשר לתוכנית שלך לעבוד עם קבוצות פקדים בזמן פעולת התוכנית. שימוש במערכי פקדים יכול להקל על ניהול קבוצות פקדים בזמן ריצת התוכנית.

מבוא למערכי פקדים

מערך פקדים (Control Array) הוא קבוצת פקדים מאותו סוג, בעלי שם זהה, המזוהים על ידי אינדקס. אם נושא המערכים מוכר לך, תמצא שמערכי פקדים עובדים בצורה דומה.

אלמנטים במערכי פקדים

ניתן להתייחס לכל פקד יחיד במערך כ**אלמנט** (Element) של המערך. כל אלמנט במערך הפקדים צריך לקיים מספר קריטריונים:

- כל הפקדים חייבים להיות מסוג זהה. מערך הפקדים יכול להכיל תוויות (Labels) או תיבות טקסט (Text Boxes), אך אינו יכול להכיל את שני הסוגים יחד.
- כל הפקדים הינם בעלי מאפיין Name זהה.
- ניתן לזהות כל פקד על ידי ערך מאפיין Index שלו. מאפיין אינדקס הינו משתנה מסוג Integer, לפיכך הערך המירבי שלו הוא 32767.

לעיתים קרובות, מאפייני התצוגה של הפקדים במערך (כגון Font ו-BorderStyle) יהיו זהים, אולם אין זה הכרחי. המאפיין היחיד שעליו להיות זהה עבור כל הפקדים במערך הוא Name. שאר המאפיינים עשויים להיות שונים עבור כל אחד מהאלמנטים.

הבנת היתרונות הטמונים במערכי פקדים

עבודה עם מערכי פקדים במקום עם פקדים בודדים מסייעת בעיצוב ממשק התוכנית ובניהול הקוד. להלן חלק מהיתרונות של מערך פקדים:

- הוספת אלמנט למערך פקדים גוזלת פחות משאבי מערכת מאשר הוספת פקד בודד מאותו הסוג. לדוגמה, שלוש תיבות טקסט בודדות (Text1, Text2 ו-Text3) תצרוכנה יותר משאבים מאשר מערך של שלושה אלמנטים מסוג תיבת טקסט.
- האלמנטים במערך פקדים חולקים מערכת משותפת של שגרות אירועים. משמעות הדבר היא שדי לכתוב את הקוד פעם אחת בלבד כדי לטפל באותו אירוע עבור כל הפקדים במערך.

❖ על ידי שימוש במערכי פקדים תוכל לעקוף את מגבלת השימוש בלא יותר מ-254 שמות פקדים בטופס אחד. באופן תיאורטי, תוכל להגדיר במערך פקדים כמות בלתי מוגבלת של אלמנטים. מכיון שמערך פקדים מהווה עבור Visual Basic שם אחד בלבד, כמות הפקדים אשר תוכל להציב בטופס כולו גדלה באופן ניכר.

אזהרה:

אל תגזים בשימוש בפקדים! הצבת יותר מדי פקדים בטופס גורמת לטעינה איטית יותר של הטופס, ועלולה להקשות על כמות משאבי המערכת.

יצירת מערך פקדים

ב- Visual Basic 5 לא ניתן היה להוסיף פקדים בזמן ריצת התוכנית, אלא כחלק ממערך קיים בלבד. ב- Visual Basic 6 ניתן להוסיף פקדים עצמאיים גם בזמן ריצה.

ניתן ליצור מערך פקדים בשלוש דרכים שונות:

❖ הוספת פקד לטופס ושימוש בפקודות Copy ו-Paste כדי לשכפלו.

❖ הוספת פקדים בודדים לטופס ואחר שינוי מאפיין Name של כל הפקדים, כך שיכילו שם זהה.

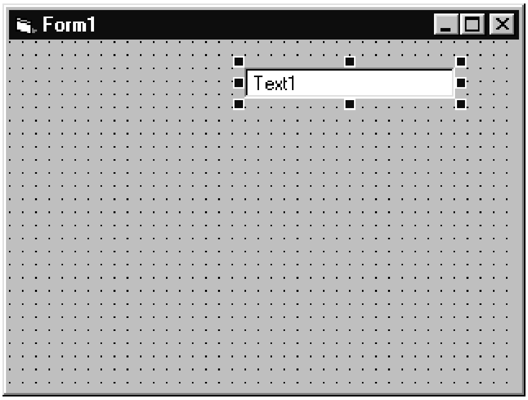
❖ הוספת פקד לטופס, ולאחר מכן הגדרת מאפיין Index שלו כמספר בין 0 ל-32767.

בסעיפים הבאים, תיצור מערך פקדים על ידי שימוש בשיטה הראשונה.

הוספת מערך פקדים לטופס

כאן, תוסיף מערך בעל ארבעה אלמנטים מסוג תיבת טקסט (Text Box), ומערך נוסף בעל ארבעה אלמנטים מסוג תווית (Label). בצע את הצעדים הבאים:

1. אם צריך, צור פרויקט חדש והוסף פקד **TextBox**, כמתואר בתרשים 13.1.
2. קבע את מאפייני תיבת הטקסט כמתואר להלן:
   ❖ קבע את מאפיין **Name** כ-**txtArray** (או כל שם אפשרי אחר).
   ❖ נקה את מאפיין **Text**.
   ❖ קבע את מאפיין Font כ-**Courier New**, גודל **10**.
3. לחץ על תיבת הטקסט כדי להבטיח שהיא עדיין במיקוד, והקש **Ctrl+C** או בחר באפשרות **Copy** מתוך תפריט **Edit** כדי להעתיק אותה ללוח (Clipboard).
4. לחץ על הטופס כדי להבטיח שהוא נמצא במיקוד. הקש **Ctrl+V** או בחר באפשרות **Paste** מתוך תפריט **Edit** כדי להציב עותק של תיבת הטקסט בטופס.
5. בנקודה זו תופיע על המסך תיבת דו-שיח אשר תבקש לוודא שבכוונתך ליצור מערך פקדים (ראה תרשים 13.2). בחר **Yes** כדי ליצור מערך שכזה.



תרשים 13.1: למרות שתיבת טקסט זו עתידה להיות חלק ממערך פקדים, יוצרים אותה בדרך הרגילה



תרשים 13.2: הדרך הקלה ליצירת מערך פקדים היא להעתיק ולהדביק פקד קיים

הבט במאפייני הפקד החדש (המודבק). חלון המאפיינים מציג את מאפייני הפקד txtArray(1), הלא הוא האלמנט החדש במערך הפקדים שזה עתה יצרת. שים לב כי מאפייני Font ו-Text של האלמנט החדש תואמים לאלה של תיבת הטקסט המקורית. בנוסף, שים לב גם כי אינדקס הפקד החדש נקבע כ-1.

6. גרור את תיבת הטקסט החדשה ומקם אותה מעל לתיבה המקורית.

7. לחץ על התיבה המקורית כדי לסמן אותה. שים לב כי מאפיין Index שלה אשר היה קודם ריק, מראה כעת את הערך 0.

8. תיבת הטקסט שהעתקת בצעד 3 נמצאת עדיין בלוח, לפיכך אין צורך להעתיקה שוב. הדבק תיבה נוספת לטופס וגרור אותה אל מתחת לשתי התיבות הקודמות.

ודא כי מאפייני הפקד החדש מתאימים לשניים שקדמו לו, וכי האינדקס שלו הוא 2. כל פקד המתווסף למערך מקבל אינדקס הגדול באחד מהאינדקס שניתן לאלמנט הקודם. מספרי האינדקס במערך פקדים מתחילים מ-0.

הערה:

תיבת הדו-שיח המבקשת לאשר את יצירת מערך הפקדים, מופיעה בעת שכפול הפקד הראשון. לאחר אישור הפעולה היא לא תופיע יותר, גם אם תבצע שכפולים נוספים במסגרת אותו מערך.

9. הוסף תיבת טקסט רביעית באותו אופן, גרור אותה כך שתמוקם מתחת לאחרים, ושנה את מאפיין **Width** שלה ל-**495** twips.

10. הוסף פקד תווית **lblArray**, ומקם אותו משמאל לתיבת הטקסט הראשונה. קבע את מאפיין **Caption** כ-**First Name:** ואת מאפיין היישור ל-**1-Right Justify**.

11. העתק את פקד התוויות ללוח והדבק עותק שלו לטופס. השב **Yes** לשאלה האם אתה רוצה ליצור מערך פקדים. קבע את **Caption** של העותק כ-**Last Name:**, וגרור את התווית לשמאל התיבה השנייה. כאמור, מלבד מאפיין Name, אשר נשאר זהה עבור כל הפקדים במערך, תוכל לקבוע את שאר המאפיינים על פי רצונך, כאילו מדובר בפקדים נפרדים.

12. צור שני עותקים נוספים של פקד התוויות. מקם אותם משמאל לתיבות הטקסט השלישית והרביעית, וקבע את מאפייני Caption שלהם כ-**City:** ו-**State:**.

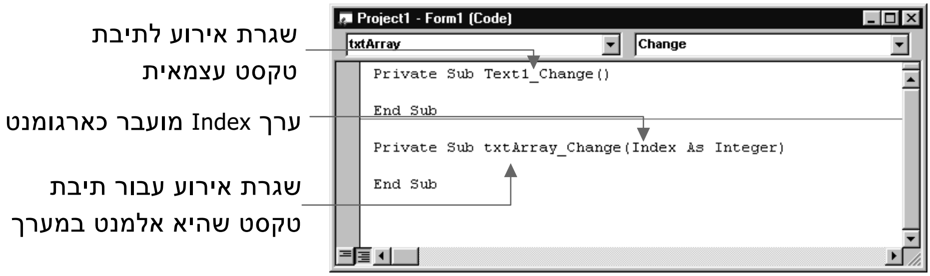
תרשים 13.3 מראה כיצד נראה הטופס לאחר השלמת העיצוב. בנוסף, התרשים מדגיש את מאפיין Index של פקד מסוים. הערך הקיים במאפיין זה מציין כי הפקד הינו חלק ממערך פקדים.



תרשים 13.3: טופס זה מכיל שני מערכי פקדים

כתיבת קוד עבור מערך פקדים

לאחר שיצרת מערך פקדים על גבי הטופס, תוכל לכתוב קטע קוד יחיד המטפל באירוע מסוים שיוצרים פקדי המערך. בעת התרחשות אירוע, Visual Basic מציבה את ערך מאפיין Index של הפקד הרלוונטי (זה שיצר את האירוע) בקוד שגרת האירוע. תהליך זה מודגם בתרשים 13.4.



תרשים 13.4: שגרות אירועים הכתובות עבור מערכי פקדים עושות שימוש במשתנה Index כדי לפעול על האלמנט שיצר את האירוע

טיפול באירועי מערכי פקדים

כתיבת קוד עבור האלמנטים במערך הפקדים נעשית דרך חלון הקוד, בדומה לכתיבת קוד עבור פקד בודד. קוד האירוע שכתבת יבוצע בכל פעם שאחד האלמנטים במערך יצור את אותו אירוע.

כעת תוכל לשפר את פרויקט הדוגמה על ידי הוספת שגרת אירוע. השיגרה שתכתוב צריכה להמיר את הטקסט שהקליד המשתמש בטופס לפורמט המתאים, כלומר להמיר את האות ראשונה בכל מילה לאות גדולה. תוכל לעשות זאת על ידי שימוש בפונקציה של Visual Basic, המכונה StrConv.

אולם, היכן עליך לכתוב את הקוד? לפקדים המסוגלים לחוש את מצב המיקוד (Focus) על גבי הטופס יש שגרת אירוע בשם LostFocus, המתבצעת כאשר המיקוד עובר מהפקד הנוכחי לפקד אחר. אפשרות זו נותנת לך הזדמנות מצוינת לשמירה על מבנה הטקסט בתיבות טקסט. תוכל להוסיף קוד לשגרת LostFocus של מערך פקדי תיבות הטקסט. קוד זה יופעל בכל פעם שתיבת טקסט במערך מאבדת את המיקוד (אירוע זה מתרחש לרוב כאשר המשתמש סיים להקליד את הטקסט בתיבה).

כדי לכתוב את שגרת LostFocus של מערך הפקדים, בצע את המהלכים הבאים:

1. לחץ לחיצה כפולה על תיבת טקסט שבמערך כדי להציג את חלון הקוד. עבור לשגרת txtArray_Change. ודא כי הפרמטר Index As Integer זמין לשיגרה.
2. אינך מעוניין לכתוב קוד בתוך שגרת Change, ולפיכך עבור לשגרת LostFocus.
3. כעת אתה אמור להימצא בשגרת txtArray_LostFocus המכילה גם היא את הפרמטר Index As Integer. הקלד את שורת הקוד הבאה לתוך השיגרה:

```
txtArray(Index) = StrConv(txtArray(Index),vbProperCase)
```

הצד השמאלי של פקודת ההצבה, txtArray(Index), מתייחס לאלמנט מסוים במערך פקדי txtArray. אולם באיזה אלמנט מדובר? מציין המערך, Index, המועבר לשגרת האירוע כמשתנה, מאפשר לנו לקבוע איזה אלמנט במערך הפקדים איבד את המיקוד וגרם לשיגרה זו להתבצע. לפיכך, שורת קוד זו מפעילה את הפונקציה StrConv על האלמנט הרלוונטי וגורמת להמרת הטקסט הנוכחי לטקסט תקין (vbPropercase).

לאחר סגירת חלון הקוד, בחר את אחד האלמנטים במערך תיבות הטקסט והכנס לחלון הקוד שלו. שים לב כי בחלון הקוד מופיעה אותה שיגרה בדיוק. **זכור, אלמנטים במערך פקדים משתמשים באותן שגרות אירועים.**

כעת תוכל לנסות את תוכנית הדוגמה. הפעל את התוכנית והקלד את הטקסט TINA MARIE באותיות גדולות בתיבת הטקסט הראשונה (ליד תווית First Name). כאשר תקיש Tab כדי לעבור לתיבה הבאה, תופעל שגרת LostFocus והטקסט שהקלדת יומר לפורמט המתאים. הקלד את המילה jones באותיות קטנות בשדה Last Name. הקש על Tab שנית ושים לב שהטקסט הזה מומר גם הוא. שגרת LostFocus שכתבת מתבצעת בכל פעם שאירוע LostFocus מתרחש עבור כל אחד מהפקדים במערך.

בידוד אלמנטים במערך פקדים בעזרת מאפיין Index

אם יש צורך לבצע קוד מסוים עבור אלמנט אחד בלבד או עבור קבוצה מסוימת של אלמנטים במערך, תוכל להשתמש במאפיין Index כדי לקבוע מיהו האלמנט שיצר את האירוע. לאחר מכן תוכל, לדוגמה, להציב את הקוד הנחוץ בבלוק If או Select Case.

בחן את העניין בעזרת פרויקט הדוגמה. שגרת האירוע שיצרת עובדת מצוין עבור מרבית האלמנטים במערך, אולם קיצורי שמות מדינות, המוקלדים בתיבת הטקסט State, צריכים להופיע כולם באותיות גדולות. לפיכך, יש צורך להתאים מחדש את שגרת txtArray_LostFocus כך שתוכל לבודד את תיבת הטקסט State (אלמנט מספר שלוש במערך), ולבצע פעולה שונה על הערך הנקלט בה.

שנה את הקוד בשגרת txtArray_LostFocus לפקודות הבאות:

```
If Index = 3 Then
	txtArray(Index) = UCase(txtArray(Index))
Else
	txtArray(Index) = StrConv(txtArray(Index), vbProperCase)
End If
```

נסה את השיגרה החדשה: הפעל את התוכנית מחדש, עבור בין תיבות הטקסט והקלד ערכים באותיות קטנות בלבד. שים לב כי שגרת האירוע ממירה את הטקסט שהוקלד בתיבת State באופן שונה משאר תיבות הטקסט שבטופס.

כלי קוד לטיפול במספר פקדים יחדיו

יתרון נוסף הנגזר משימוש במערך פקדים הוא היכולת לקבוע את מאפייני הפקדים במהלך ביצוע התוכנית על ידי שימוש בלולאת For...Next. בדרך זו, תוכל להתייחס לכל הפקדים במערך צורה נוחה ויעילה. לדוגמה, שורות הקוד הבאות ממירות את הטקסט שנקלט בכל האלמנטים במערך פקדים לאותיות גדולות:

```
Dim I As Integer
For I = 0 To 3
	txtArray(I) = UCase(txtArray(I))
Next I
```

שים לב כי לולאת For...Next מונה מ-0 ל-3 ולא מ-1 ל-4 מפני שהאלמנטים במערך ממוספרים בדרך כלל מ-0.

הסרת אלמנטים ממערך פקדים

בסביבת העיצוב תוכל להסיר אלמנטים ממערך פקדים בטופס במספר דרכים:

- על ידי בחירת האלמנט הרצוי והקשת Delete.
- על ידי בחירת האלמנט, לחיצה על תפריט Edit ובחירת האפשרות Delete.
- על ידי לחיצה ימנית על האלמנט ובחירת האפשרות Delete מהתפריט המקוצר.

בצורה זו, תוכל למחוק את הפקד מהטופס ולהוריד את כמות האלמנטים במערך הפקדים. אם מסיבה כלשהי, תרצה לשנות את האלמנטים במערך פקדים כך שלא יהיו שייכים למערך, עליך להגדיר לכל אחד מהם (פרט לאחד בודד) שם חדש, ולמחוק את ערך מאפייני Index שלהם כך שיחזרו להיות חסרי ערך (Null).



אזהרה:

בהסרת פקד מהמערך, שאר הפקדים אינם ממוספרים מחדש בצורה אוטומטית. משום כך, מחיקת פקדים עלולה להשאיר חללים במספור ערכי Index של האלמנטים שנותרו במערך. חללים אלה עלולים ליצור בעיות אם לדוגמה קיים קוד המשתמש בלולאת For כדי לעבוד עם האלמנטים במערך. במקרה זה, תתרחש שגיאה כאשר הלולאה תנסה לגשת לאלמנט במערך על ידי שימוש בערך Index שנמחק.

עבודה עם מערכי פקדים

כעת, לאחר הבנת המכניזם העומד מאחורי יצירת מערכי פקדים, תלמד מספר דרכים לשימוש בו. סיבה אחת לאיחוד קבוצת פקדים כמערך אחד, הם כאשר נרצה להפעיל קוד זהה לעיבוד הנתונים על כל אחד מהפקדים בקבוצה. דוגמה לכך הן ארבע תיבות טקסט המייצגות יחדיו פרטי כתובת. סיבה נוספת היא פישוט כתיבת הקוד עבור שגרות אירועים. לדוגמה, אם יש לך 26 לחצני פקודה המייצגים את אותיות האלף-בית הלועזי, קל יותר לטפל בהם כמערך מאשר להתייחס לכל פקד בנפרד.

שימוש במערכי פקדים בתוכניות

כפי שכבר למדת, קיימים כמה שימושים למערכי פקדים בתוכניות. אחד מהם הודגם קודם לכן, כאשר המרת טקסט, בכל אחד מהפקדים שעל הטופס לאותיות גדולות.

פעולה נפוצה נוספת, במסגרתה נעשה שימוש במערכי פקדים, היא איתור לחצן האפשרויות (Option Button) שנבחר מתוך מערך לחצנים. מאחר וניתן לבחור לחצן אחד בלבד בכל זמן נתון, תוכל להשתמש בלולאת For, כפי שמציגה הדוגמה הבאה:

תוכנית דוגמה 13.1: OPTIONARRAY.TXT - שימוש במאפייני Lbound ו-Ubound כדי לקבוע את טווח מערך הפקדים

```
Dim I As Integer, nChosen As Integer
For I = optArray.Lbound To optArray.Ubound
	If optArray(I).Value Then
		nChosen = I
		Exit Sub
	End If
Next I
```

כפי שבוודאי הבחנת, תוכנית 13.1 משתמשת במאפיינים Lbound ו-Ubound במקום להגדיר ערכים מדויקים עבור לולאת For. מאפיינים אלה מחזירים את הגבול התחתון והעליון (בהתאמה) של ערכי האינדקס במערך הפקדים. באמצעות מאפיינים אלה ניתן לבצע הוספה והסרה אוטומטית של פקדים ממערכים. מאפיינים אלה מקושרים למערך עצמו ולא לאלמנטים שבו.

הערה:

מאפייני Lbound ו-Ubound מספקים ערכי מינימום ומקסימום של מספרי האינדקס במערך, אך אין בכך כדי להצביע שהאלמנטים במערך מסודרים בסדר רציף.

מערכים מקבילים

טכניקה שימושית במערכי פקדים היא שימוש במספר מערכים המכילים אלמנטים בעלי מכנה משותף. מערכים אלה נקראים **מערכים מקבילים** (Parallel Arrays).

ניקח כדוגמה את שני מערכי הפקדים שיצרת בתוכנית הדוגמה. מערך אחד הוא קבוצת תיבות טקסט אשר לתוכה המשתמש יכול להזין טקסט. המערך השני הוא קבוצת פקדי תוויות, כאשר כל תווית מתאימה לתיבת טקסט במערך הראשון. נניח שתרצה לשנות את התוכנית כך שתדגיש את התווית השייכת לתיבת הטקסט הנמצאת במיקוד, כך שתהווה סימן מזהה למשתמש. תוכל להשיג מטרה זו על ידי הוספת קוד לשגרת GotFocus (שיגרה המבוצעת בעת שהפקד מקבל מיקוד) של מערך תיבות הטקסט. קוד זה יקבע את מאפיין Font.Bold של התווית הרצויה. כלומר, כאשר התוכנית מתמקדת בתיבת טקסט מסוימת, יגרום הדבר גם להדגשת התווית המשויכת אליה. כן יש להוסיף קוד לשגרת LostFocus (שיגרה המבוצעת בעת שהפקד מאבד מיקוד) של מערך תיבות הטקסט אשר יחזיר את התווית למצב רגיל.

כדי להשיג מטרה זו, בצע את הצעדים הבאים:

1. הקלד את שורת הקוד הבאה לתוך שגרת GotFocus של מערך txtArray:

```
lblArray(Index).Font.Bold = True
```

שים לב כי הקוד, למרות שהוא נמצא בשגרת אירוע של מערך פקדי תיבות הטקסט, פועל למעשה על פריט במערך התוויות המקביל. משתנה Index משמש לקביעה על איזה מהאלמנטים במערך התוויות יש לבצע את הפעולה.

2. הוסף את השורה הבאה לסוף שגרת txtArray_LostFocus:

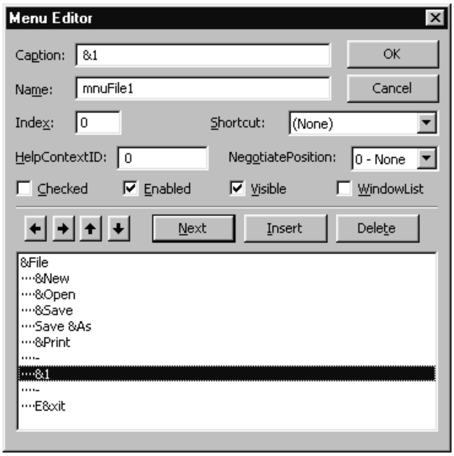
```
lblArray(Index).Font.Bold = False
```

3. הרץ את התוכנית. כאשר תעבור בין תיבות הטקסט, שים לב כי התווית המשויכת לתיבה הפעילה מודגשת.

יצירת מערך פריטי תפריטים

בנוסף למערכי פקדים, תוכל ליצור מערך המורכב מפריטי תפריט. כמו במערך פקדים, מערכי תפריטים הם הדרך היחידה בה תוכל להוסיף פריטים לתפריט מסוים בזמן פעולת התוכנית. תוכל להשתמש במערך תפריטים כדי להציג את רשימת הקבצים האחרונים שהיו בשימוש על ידי התוכנית. אפשרות זו שכיחה בתוכניות רבות.

כדי ליצור מערך תפריטים, יש צורך לפתוח את עורך התפריטים (לחץ על לחצן Menu Editor או בחר באפשרות Menu Editor מתוך תפריט Tools). לאחר מכן, צור פריטי תפריט על ידי הזנת מאפייני Caption ו-Name עבור כל אחד מהפריטים. תרשים 13.5 מציג את עורך התפריטים אשר בו מוגדרים המאפיינים עבור מה שיכול להיות פריט ראשון ברשימת הקבצים האחרונים בשימוש.



תרשים 13.5: תוכל להגדיר את מאפיין Index של פקד תפריט מתוך עורך התפריטים כדי ליצור מערך המורכב מפקדי תפריטים

בדומה למערכי פקדים, אלמנטים במערך תפריטים חייבים לעמוד בדרישות מסוימות:

- כל האלמנטים צריכים להיות בעלי מאפיין Name זהה.
- כל האלמנטים צריכים להשתייך לאותה רמה בתפריט.
- הפריטים במערך חייבים להופיע ברציפות, אחד אחרי השני בתפריט. אם תרצה להוסיף שורת הפרדה בתוך הרשימה, עליך לכלול אותה כחלק מן המערך.
- לכל אלמנט במערך חייב להיות מספר אינדקס ייחודי לו.

טעינה והסרה בעת ריצה

בפרקים הקודמים בספר למדת כיצד להוסיף פקדים לטפסים בסביבת העיצוב. בסעיפים האחרונים למדת כיצד ליצור מערכי פקדים כדי לטפל בקבוצות פקדים בצורה נוחה. אולם, עדיין לא למדת כיצד להוסיף ולהסיר פקדים בעת ריצת התוכנית.

טיפ:

אם אתה מתכנן להוסיף פריטי תפריט בזמן פעולת התוכנית, הצב את מערך התפריטים בסוף התפריט. באופן זה, תגרום לתפריט להתרחב אחרי הפריטים הקבועים ובכך לא תבלבל את המשתמש על ידי שינוי מיקום האפשרויות בתפריט.

הערה:

תוכל להסתיר ולהציג פקדים בזמן פעולת התוכנית על ידי שינוי ערך מאפיין Visible של הפקד המבוקש. אולם, עליך ליצור את הפקדים המבוקשים לפני שתוכל להשתמש בהם. הסתרה והצגה של פקדים אינה זהה להוספתם למערך.

אזהרה:

אם קיימות פקודות If או Select המתייחסות לפקדים לפי ערך האינדקס, ודא שהקוד פועל על כל האלמנטים במערך, כולל אלמנטים חדשים אשר עלולים להתווסף בעת ריצה. מאפייני Lbound ו-Ubound מסייעים להשיג מטרה זו. אך זכור שהאלמנטים במערך לא מסודרים בהכרח בסדר רציף.

יצירת האלמנט הראשון במערך פקדים

אנו ניצור את האלמנט הראשון שלנו בזמן פיתוח כדי לפשט את הדוגמה:

1. הוסף את הפקד לטופס.
2. קבע את מאפיין Name של הפקד.
3. קבע את מאפיין Index של הפקד כמספר שלם (אפס בדרך כלל).

שים לב כי הכנסת ערך כלשהו במאפיין Index גורמת ליצירת מערך פקדים. אם תעתיק ותדביק פקדים, Visual Basic תקבע את ערך המאפיין עבורך.

הוספת פקדים בזמן פעולת התוכנית

לאחר שיצרת את האלמנט הראשון במערך, תוכל להשתמש בקוד כדי להוסיף אלמנטים אחרים בתגובה לאירועים אשר יתרחשו. ביישומים רבים תוכל להוסיף את הפקדים בזמן טעינת הטופס. דוגמה לשימוש בטכניקה זו הינה ליצור טופס המיועד לקליטת מידע כללי. על ידי שימוש במערך פקדים, תוכל להעביר קבוצה ראשונית של רשומות לטופס, ולאחר מכן ליצור פקדים בכמות שתרצה, כדי לטפל במספר משתנה של שדות ברשומה. (מערך רשומות (Recordset), מייצג מידע בבסיס נתונים. מידע נוסף על רשומות ניתן למצוא בפרק 24 **יסודות מסד הנתונים**).

כדי להוסיף פקד בזמן פעולת התוכנית, עליך להשתמש בפקודת Load. פקודה זו זהה לפקודה המעלה טופס לזיכרון. כאשר אתה משתמש בפקודה, עליך לכלול את שם הפקד הרצוי ואת ערך האינדקס שלו. כדי למנוע טעויות, עליך לוודא שערך האינדקס ייחודי. שים לב כי מאפיין Visible של הפקד נקבע כ-False. להלן תהליך הוספת פקד:

1. השתמש בפקודת Load כדי ליצור את הפקד החדש ולטעון אותו לזיכרון.
2. הפקד החדש יוצג במיקום הפקד המקורי מפני שהוא יורש את מאפייני הפקד המקורי (כולל ערכי המיקום שלו). לפיכך, עליך להזיז את הפקד בעזרת פקודת Move או על ידי קביעת ערכי מאפייני Top ו-Left שלו.
3. כדי שהפקד יופיע על הטופס, קבע את מאפיין Visible כ-True.
4. קבע מאפיינים נוספים לפי הצורך (כל ערכי המאפיינים הועברו מהפקד שעוצב במקור בסביבת העיצוב).

טיפ:

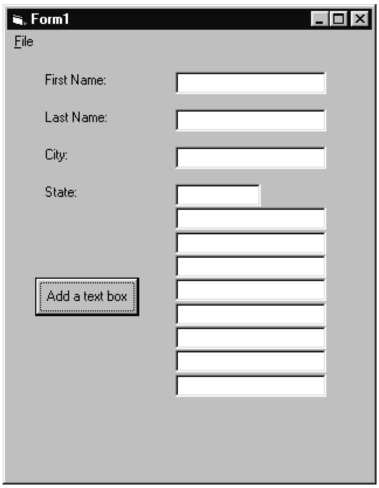
קביעת ערך האינדקס של האלמנט החדש, כאחד אחרי מאפיין **Ubound**, מבטיח שלאלמנט ייוּתן ערך אינדקס ייחודי.

כעת, בדוק את הטכניקה על ידי הוספתה לתוכנית הדוגמה. הוסף לחצן בדיקה שיהיה אחראי על הוספת אלמנט למערך פקדי txtArray על ידי ביצוע המהלכים הבאים:

1. בסביבת העיצוב, קבע את מאפיין **Height** של הטופס הראשי כ-**6000**. ערך זה מאפשר מקום רב עבור הפקדים החדשים אשר יתווספו כאשר התוכנית תרוץ.
2. הוסף לחצן פקודה (Command Button) באזור ריק בטופס. קבע את שמו כ-**cmdAdd** ואת מאפיין **Caption** שלו כ-**Add a text box**.
3. הזן את הקוד הבא לתוך שגרת Click של מערך cmdAdd:

```
Dim I As Integer
I = txtArray.Ubound
Load txtArray(I + 1)
txtArray(I + 1).Visible = True
txtArray(I + 1).Top = txtArray(I).Top + txtArray(I).Height + 120
txtArray(I + 1).Left = txtArray(I).Left
```

הקוד המובא כאן קובע את מיקום האלמנט החדש במערך כך שיהיה 120 נקודות (twips) מתחת לפקד האחרון שהתווסף (תוך התחשבות בגובה הפקד עצמו). הפעל את התוכנית ולחץ על הלחצן כדי לנסות אותה. תרשים 13.6 מראה כיצד ייראה הטופס לאחר מספר לחיצות.



תרשים 13.6: ניתן להוסיף אלמנטים למערך פקדים בזמן פעולת התוכנית

הסרת פקדים בזמן פעולת התוכנית

כשם שאתה נדרש להוסיף פקדים במהלך ביצוע התוכנית, תידרש גם להסיר אחד או יותר מהם. ניתן להסיר פקד בעזרת פקודת Unload. כמו בפקודת Load, יש צורך לציין את שם הפקד ואת ערך האינדקס שלו כמתואר בדוגמה הבאה:

```
Unload txtArray(5)
```

אזהרה:

תוכל להשתמש בפקודת Unload כדי להסיר פקדים אשר נוצרו בזמן פעולת התוכנית בלבד, ולא להסרת פקדים אשר נוצרו קודם לכן בסביבת העיצוב.

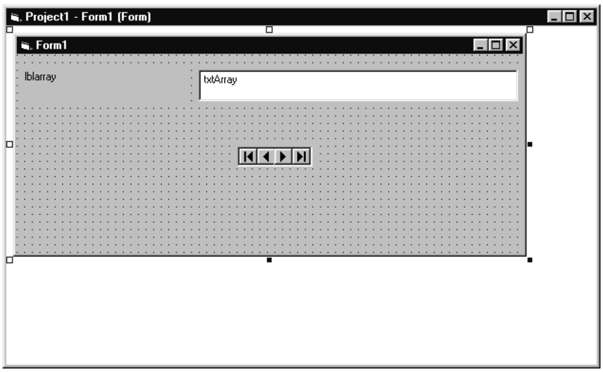
קודם לכן גילית שניתן להשתמש במערכי פקדים כדי ליצור טופס לקליטת מידע כללי. מטרת הטופס היא לבדוק את מבנה הרשומה הרצויה, וליצור פקדים המטפלים בכל השדות ברשומה. בסעיף זה תלמד כיצד פועל טופס מסוג זה.

כדי לפשט את העניין, נגדיר עבור הטופס שני מערכים: הראשון עבור תוויות המתארות את השדות, והשני עבור תיבות הטקסט אשר תקבלנה את ערכי השדות ברשומה. כדי להתחיל את הגדרת הטופס, הוסף מערך תוויות ומערך תיבות טקסט לטופס כמתואר בתרשים 13.7.

הערה:

כדי ליצור אלמנטים במערך ולא פקדים נפרדים, קבע את ערכי האינדקס של התוויות ותיבת הטקסט כ-0.

בנוסף, יש צורך להוסיף **פקד נתונים** (Data Control) לטופס כדי לספק את מבנה הרשומה לטופס. תוכל ללמוד על פקד הנתונים ופקדים הקשורים לו בפרק 25 **פקד הנתונים ופקדי איגוד נתונים**.



תרשים 13.7: בסביבת העיצוב, טופס קליטת המידע בגרסתו ההתחלתית, מכיל את האלמנטים הראשונים בשני מערכי הפקדים

הערה:

DataBaseName - המאפיין שמכיל את שם DataBase (מסד הנתונים).
RecordSource - המאפיין שמכיל את שם מקור הנתונים מתוך מסד הנתונים.

עליך לקבוע את מאפייני DatabaseName ו-RecordSource של פקד הנתונים כדי לגשת לרשומה. לאחר שהגדרת את פקד הנתונים, הקצה את הפקד למאפיין DataSource של תיבת הטקסט. מאפיין זה יועבר לכל אחת מתיבות הטקסט שתתווספנה לטופס. לאחר מכן הזן את הקוד המובא בתוכנית 13.2 לטופס.

תוכנית 13.2: SAMPLE2.VBP - שימוש באוסף השדות לצורך יצירת פקדים

```
Option Explicit

Dim bFirstLoad As Boolean
Private Sub Form_Activate()
    Dim I As Integer
    Dim nFields As Integer
    Dim sName As String
    If Not bFirstLoad Then Exit Sub

    With dtaMain.RecordSet
        nFields =.Fields.Count
        For I = 0 To nFields - 1
            sName =.Fields(I).Name
            If I > 0 Then
                Load lblArray(I)
                Load txtArray(I)
                txt(I).Top = txtArray(I - 1).Top + txtArray(I - 1).Height / 120
                txtArray(I).Left = txtArray(I - 1).Left
                lblArray(I).Top = txtArray(I).Top
                lblArray(I).Left = lblArray(I - 1).Left
                txtArray(I).Visible = True
                lblArray(I).Visible = True
            End If
            lblArray(I).Caption = sName
            lblArray(I).DataField = sName
        Next I
    End With
    I = txtArray.Ubound
    dtaMain.Top = txtArray(I).Top + txtArray(I).Height + 120
    Me.Height = dtaMain.Top + dtaMain.Height + 525
    bFirstLoad = False
End Sub
```

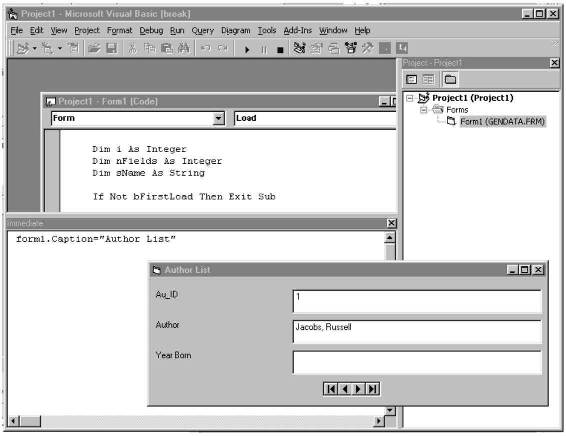
```
Private Sub Form_Load()
    bFirstLoad = True
End Sub
```

תוכנית דוגמה היא Sample2.VBP.

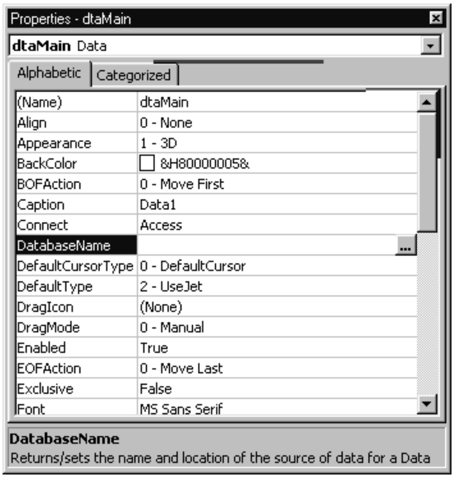
קוד זה מוסיף תווית חדשה ותיבת טקסט עבור כל שדה אחרי השדה הראשון. פקדים חדשים מוצבים מתחת למערכת הפקדים האחרונה. לאחר מכן נקבעים מאפייני Caption של פקד התווית ו-DataField של פקד תיבת הטקסט. לאחר שנתווספו כל השדות, התוכנית מציבה את פקד הנתונים מתחת לפקדים האחרים, ומשנה את גודל הטופס כך שיכיל את כל הפקדים. דוגמה לכך מוצגת בתרשים 13.8.

הערה:

בטופס מוגדר מסד הנתונים Biblio.mdb כממוקם ב- C:\VB6\Biblio.mdb. מסד נתונים זה הינו מסד נתונים לדוגמה שמגיע עם Visual Basic. אם מיקומו שונה אצלך במחשב, שנה את מיקומו במאפיין DataBaseName, ראה תרשים 13.9.



תרשים 13.8: טופס הנתונים הכללי יכול לטפל כמעט בכל סוג רשומה



תרשים 13.9: מאפיין DataBaseName של מסד הנתונים

מכאן...

בפרק זה, למדת מספר טכניקות בהן תוכל להרחיב את ביצועי תוכניותיך באמצעות שימוש בפקדים. למדת כיצד לשפר את יכולות התכנות שלך על ידי שימוש במערכים המקבצים יחדיו קבוצת פקדים בעלי מכנה משותף. בנוסף, ראית כיצד להוסיף פקדים בצורה גמישה במהלך ביצוע התוכנית.

למידע נוסף על חלק מהנושאים המדוברים בפרק זה, עיין בפרקים הבאים:

- כדי לקבל מידע כללי נוסף על פקדים, ראה פרק 4 **שימוש בפקדי ברירת המחדל של Visual Basic**, ופרק 12 **הפקדים המשותפים של Microsoft**.
- בסיסי נתונים ורשומות מוצגים בפרק 24 **יסודות מסד הנתונים**, ומתוארים בצורה מפורטת יותר בפרק 26 **שימוש באובייקטי גישה לנתונים (DAO)**.

14

יצירת פקדי ActiveX

מה בפרק זה?

- מבוא ל-ActiveX
- יצירת פקד ActiveX
- בדיקת פקד ActiveX
- תרגום הפקד (הידור, Compilation)
- שיפור פקד ActiveX
- שימוש באשף הממשקים עבור פקדי ActiveX
- שימוש באשף דפי המאפיינים
- יצירת פקד ActiveX מותאם אישית

בפרקים הקודמים למדת כיצד ליצור שילובי פקדים הקיימים ב- Visual Basic כדי לבנות יישום חזק. אחת האפשרויות המלהיבות ביותר של Visual Basic היא האפשרות ליצור פקדים משלך. פקדים מותאמים אישית אלה, הידועים גם בשם **פקדי ActiveX** (ActiveX Controls), יכולים לשמש כפקדים בכל יישום Visual Basic. (**ActiveX** הינו כינוי של חברת Microsoft לקבוצת פריטים הכוללת פקדים, ספריות DLL ומסמכי ActiveX). בנוסף ליתרון הברור בשימוש בקוד, ניתן להשתמש בפקדי ActiveX בדף Web כדי לקבל תפקוד הדומה לפעולת תוכנית באינטרנט.

בפרק זה תראה את הגישות שתוכל לנקוט וכמה מהדברים שעליך לקחת בחשבון כאשר אתה יוצר פקדי ActiveX. המידע הקיים בפרק זה יורחב בפרק הבא, **הרחבת פקדי ActiveX** אשר ילמד אותך כיצד לבנות פקדים מתוחכמים יותר מסוג זה.

מבוא ל-ActiveX

כעת אתה כבר מכיר פקדים כגון פקד TextBox ופקדי תוויות. כדי להשתמש בפקדים אלה, עליך לשרטט אותם על טופס ולשלוט בצורה שבה הם יתנהגו בעזרת מאפייניהם, השגרות השייכות להם והאירועים המקושרים אליהם. כאשר אתה יוצר פקד ActiveX משלך, אתה יוצר אובייקט דומה, אולם אתה קובע מה יהיו מאפייניו, השגרות השייכות לו והאירועים אשר יכולים להפעיל אותו. לאחר יצירת הפקד, תוכלו אתה ואחרים להשתמש בו בפרויקטים של Visual Basic, בדיוק כפי שאתה משתמש בפקדים אחרים כגון פקד TextBox. למעשה, שימוש בפקד ActiveX מהווה את הדרך הטובה ביותר ליצור רכיב אשר ניתן יהיה להשתמש בו מתוך קובץ EXE נפוץ מסוג שרת/לקוח וכן על גבי האינטרנט.

הצעדים בבניית פקדי ActiveX

יצירת פקד ActiveX בשפת Visual Basic שונה מיצירת יישום EXE רגיל. הנה תקציר הצעדים הדרושים לבניית פקד כזה:

1. החלט, ברמה פרטנית, מה יהיה השימוש בפקד שתיצור. פקד ActiveX דומה לאובייקט עצמאי ולכן עליך לשאול את עצמך לאיזו מטרה אתה יוצר אותו? איך אתה רוצה להציג אותו על המסך? אילו מאפיינים, שגרות ואירועים המקושרים לו יהיו זמינים לתוכנית?
2. החלט אם תשתמש בפקדים אחרים כאבני בניין לפקד שלך. לדוגמה, תוכל ליצור פקד ActiveX אשר יכיל תצוגת רשת מפקד של חברה אחרת. כאשר אתה משתמש בפקדים חיצוניים, קח בחשבון את נושא זכויות היוצרים על הפקד.
3. התחל פרויקט חדש מסוג ActiveX Control ושרטט את הממשק עבורו.
4. הוסף קוד כדי ליצור מאפיינים, שגרות ואירועים שתרצה שיהיו לפקד.
5. בנה פרויקט ניסיון ובחן את הפקד מתוך Visual Basic. ודא שהשתמשת בכל המאפיינים, שגרות ואירועים שניתן להשתמש בהם עבור הפקד.

6. הדר את הפקד לקובץ OCX ובצע את הבדיקות על גרסת הקובץ. אם אתה מפתח את הפקד עבור שימוש באינטרנט, בדוק את הפקד כשהוא מוצב על דף ב-Web.

7. השתמש באשף **Package and Deployment** או בתוכנית שירות אחרת, כדי לבנות גירסה הניתנת להפצה המכילה את כל הקבצים הדרושים לפקד.

אסטרטגיות פיתוח

בניית פקד ActiveX ב- Visual Basic יכולה להיות משימה מורכבת או פשוטה בהתאם לבחירתך. רמת הקושי תלויה בשאלה אם אתה מתכוון להשתמש בפקדים קיימים, מה רמת תחכום הממשק, וכמובן, בכמה קוד אתה מתכוון להשתמש. בכל מקרה, תוכל לבצע את תהליך הבנייה בשלוש דרכים בסיסיות:

- **בנה את הפקד מפקדים קיימים.** גישה זו ידועה גם כבניית פקד מתוך **פקדים קיימים** (Constituent Controls). דרך זו היא הקלה ביותר מפני שאתה מאחד למעשה פקדים קיימים. לדוגמה, תוכל לארוז פקד Combo Box ו- Text Box יחד, ולכתוב קוד מינימלי אשר יגרום להם לתקשר ביניהם בדרך הרצויה.

- **שפר פקד קיים.** תוכל להשתמש בפקד קיים כנקודת הפתיחה ולשנות את מאפייניו לצרכיך. לדוגמה, תוכל לקחת פקד Text Box ולהוסיף מאפיין אישי בשם TextCaps שיגרום לטקסט להיות מוצג תמיד באותיות גדולות. שיטה זו הינה דרך טובה להשיג מהפקד את הפעולה המבוקשת בצורה מדויקת.

- **צור ושרטט פקד מותאם אישית.** תוכל לשרטט את הממשק בעצמך בעזרת פקודות גרפיות, וליצור פקד מקורי לחלוטין אשר אינו מכיל אף פקד קיים. תהליך זה דורש עבודה רבה יותר משתי השיטות הקודמות, אולם הוא מאפשר ליצור ממשק משתמש מקורי או שונה במקצת מהממשק הקבוע.

פרק זה יציג בפניך את עולם יצירת פקדי ActiveX. תלמד על כל אחת מהשיטות המתוארות כאן בעזרת פרויקטים לדוגמה, וכן תלמד כיצד להפיץ ולהדר את הפקד הסופי. פרק 15 **הרחבת פקדי ActiveX**, מציג בפניך כמה מהנושאים המתקדמים יותר לגבי פקדי ActiveX.

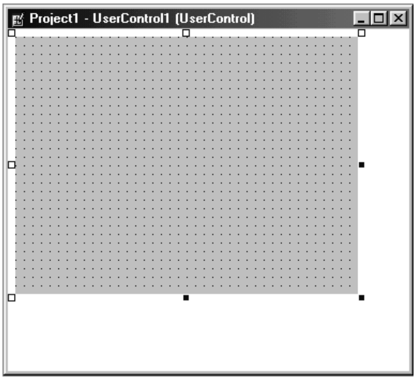
יצירת פקד ActiveX

בניית פקד ActiveX מפקדים קיימים צופנת בחובה יתרונות רבים. קודם כל, כאשר אתה מוסיף פקד קיים לפקד שלך, אתה מקבל את יכולת הפעולה המלאה של הפקד הקיים - שיקול חשוב מבחינת כמות השגרות והמאפיינים הנתמכים על ידי פקד ממוצע. שנית, שילוב מספר פקדים לפקד אחד מאפשרת להתייחס אליהם כיחידה בודדת, דבר העשוי להקל מנקודת מבט היישום. גישה זו מאפשרת גם לצייר את כל הפקדים על המסך בבת-אחת. יתרון נוסף הוא שלמשתמשים יהיה קל יותר להבין פקד הבנוי מפקדים המוכרים להם.

התחלת פרויקט פקד הדוגמה Address

כעת, לאחר שיש לך קצת רקע בנושא פקדי ActiveX, תוכל להתחיל לתרגל על ידי יצירת פקד ActiveX ראשון. הפקד שתבנה בסעיף זה יהיה פקד AddressCtl, המכיל אוסף תיבות טקסט המאפשרות למשתמש להקליד את שמו וכתובתו.

דוגמה זו תספק לך דרך טובה להבין את הנושאים הכלולים בפיתוח פקדי ActiveX בסביבת Visual Basic. פקד AddressCtl ייבנה כולו מפקדי תוויות ותיבות טקסט. ראשית, שרטט את ממשק המשתמש של הפקד והוסף קוד מינימלי כדי לאפשר לפקד לעבוד בסביבת העיצוב של Visual Basic.



תרשים 14.1: אובייקט UserControl דומה לטופס, אולם אינו מכיל שוליים או אלמנטים המופיעים בחלון רגיל

כדי לבנות את פקד Address, בצע את הפעולות הבאות.

1. היכנס לסביבת הפיתוח של Visual Basic ובחר **New Project** מתפריט **File**.
2. מתוך תיבת הדו-שיח בחר **ActiveX**. דבר זה יגרום ליצירת פרויקט חדש המכיל אובייקט UserControl בשם UserControl1, כמתואר בתרשים 14.1.
3. שנה את שם הפקד ל-**AddressCtl** בחלון המאפיינים.
4. בחר **Project Properties** מתפריט **Project** כדי להציג את מאפייני Project1.
5. ב- **Project Name**, שנה את שם הפרויקט (Project1) ל-**Address**.
6.  הוסף חמישה פקדי Text Box. מחק את מאפיין Text שלהם וקרא להם:

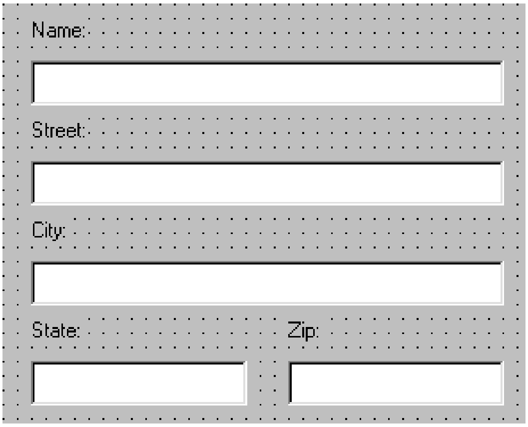
- txtName
- txtStreet
- txtCity
- txtState
- txtZip

7. הוסף חמישה פקדי תוויות לטופס ותן להם את ערכי המאפיינים הבאים:

מאפיין Caption	**מאפיין Name**
Name:	lblName
Street:	lblStreet
City:	lblCity
State:	lblState
Zip:	lblZip

8. סדר את התוויות ליד תיבות הטקסט המתאימות. לחץ על אובייקט UserContol ושנה את גודלו כך שהפקדים הנמצאים יתאימו לגודל האובייקט. תוכל לשנות את גודל הפקד על ידי שימוש בידיות שינוי הגודל או שינוי מאפייני Height ו-Width בחלון המאפיינים. פקד הדוגמה אמור להיות בגודל של כ-4395 על 3600 Twips והוא אמור להיראות כפי שמתואר בתרשים 14.2.

9. כמו שעשית בכל הפרויקטים בשפת Visual Basic, שמירת עבודתך הינה דבר חשוב ולכן עליך לעשות זאת לפני שתמשיך.



תרשים 14.2: הקבוצה המלאה של הפקדים המאוחדים משורטטת על גבי פקד UserControl

הוספת קוד לשינוי גודל הפקד

כפי שכבר ידוע לך משימוש בפקדים רגילים כגון פקד TextBox, תוכל לקבוע את גודל הפקד בסביבת העיצוב על ידי שימוש בעכבר או על ידי קביעת מאפייני Height ו-Width שלו. פקד TextBox מתוכנן להגיב לאירועים מסוג זה על ידי שרטוטו על גבי המסך בהתאם לגודלו החדש. כאשר אתה מעצב פקדי ActiveX משלך ובייחוד פקדים הבנויים מפקדים קיימים, יש לשים דגש על נושא שינוי גודלם. כאשר אתה משנה את גודל הפקד, עליך לנקוט בצעדים המבטיחים כי הפקדים הכלולים בו לא יוסתרו מעיני המשתמש או יסודרו בצורה שתמנע את פעולתו היעילה של הפקד.

כדי לטפל בפקד כזה, עליך להוסיף קוד שיידע להגיב לשגרת Resize של אובייקט UserControl. קוד זה יופעל בכל פעם שמישהו ישרטט פקד זה בפרויקט Visual Basic. כדי להוסיף את הקוד הנחוץ לפקד Address, בצע את הפעולות הבאות:

1. לחץ לחיצה כפולה על אובייקט UserControl, כפי שאתה לוחץ בדרך כלל על טופס רגיל. דבר זה יגרום לפתיחת חלון הקוד.

2. בחר את שגרת Resize מתוך תיבת השגרות של חלון הקוד.

3. הוסף את שורות הקוד הבאות לשגרת UserControl_Resize:

```
Private Sub UserControl_Resize()
    With UserControl

        ' Enforce minimum dimensions
        If .Height < 3615 Then .Height = 3615
        If .Width < 2175 Then .Width = 2175

        'Resize objects on the control
        txtName.Width = .ScaleWidth – 500
        txtStreet.Width = .ScaleWidth – 500
        txtCity.Width = .ScaleWidth – 500
        txtZip.Width = .ScaleWidth / 2 – 500
        txtState.Width = .ScaleWidth / 2 – 500
        'Move the Zipcode text box
        txtZip.Left = .ScaleWidth / 2 + 160
        lblZip.Left = .ScaleWidth / 2 + 160
    End With
End Sub
```

4. שמור את השינויים שביצעת בפרויקט.

הערה:

המספרים המגדירים את גודל הפקדים בשגרת הדוגמה נקבעו בצורה שרירותית. מטרתם לשמור על תצוגה נאה של ממשק הפקד בעת שהמפתח משנה את גודלו.

כאשר המשתמש משנה את מימדי פקד Address, שגרת Resize מופעלת. שתי שורות הקוד הראשונות בדוגמה שהובאה כאן מונעות מהמשתמש מלהפוך את עותק הפקד לקטן מדי. השורות האחרות דואגות לשנות את גודל כל הפקדים בקבוצה תוך התחשבות בגודל אובייקט Usercontrol.

הוספת מאפיין חדש לפקד

קיבוץ כמה תיבות טקסט בפקד ActiveX מציג אותן בצורה מאוחדת ומאפשר לך לשרטט את כל תיבות הטקסט בטופס בפעולה בודדת. אולם, כדי שפקד ActiveX יהיה יעיל, עליך לגשת למאפייניו, שיטותיו (**Methods**) והאירועים המקושרים אליו באותה צורה שבה הינך ניגש לפקד בודד. במילים אחרות, אם אתה מתכוון להתייחס לפקדים הכלולים בו בצורה נפרדת כאשר אתה כותב את הקוד, לא עשית הרבה כשאיחדת אותם בפקד בודד. במקרה פקד Address, מטרת הפקד היא לספק דרך לקבל את מידע הכתובת. מסיבה זו, תיצור מאפיין בשם AddressText בפקד Address, אשר יהיה דומה לפקד Text בפקדי תיבות הטקסט. ההבדל הוא שמאפיין AddressText מחזיר את הטקסט **מכל** תיבות הטקסט בטופס כמחרוזת בודדת.

כדי לשמור על פשטות העניין, המאפיין יוגדר כניתן לקריאה בלבד על ידי שימוש בשגרת המאפיין Get (מידע נוסף על שגרות מאפיינים ניתן למצוא בפרק *16*, **מחלקות: שימוש חוזר ברכיבים**).

ראה: "שגרות מאפיינים", פרק 16.

כדי לקבוע את המאפיין, פתח את חלון הקוד עבור אובייקט UserControl והוסף את הקוד הבא:

```
Public Property Get AddressText() As String
	Dim s As String

	s = txtName & vbCrLf
	s = s & txtStreet & vbCrLf
	s = s & txtCity & vbCrLf
	s = s & txtState & vbCrLf
	s = s & txtZip
	AddressText = s
End Property
```

קוד שגרת המאפיין Get אינו מתוחכם. הוא משלב את תוכן הטקסט של קבוצת הפקדים ומחזיר אותו כמחרוזת בודדת לתוכנית שקראה לו. אולם, ביצירת מאפיין זה הודגם רעיון לקיחת משימה (שילוב שדות הכתובת למחרוזת אחת) וביצועה תוך הוספת האלמנטים שלה לאובייקט (הפקד ActiveX). היתרון הוא שמפתחים אחרים יוכלו להשתמש במאפיין זה ללא צורך בהבנת דרך הפעולה של מרכיביו.

בדיקת פקד ActiveX

פקד ActiveX מגיע ללקוח בדרך כלל בצורת קובץ מהודר מסוג OCX. קובץ זה, כאשר הוא נקלט במחשב משתמש אחר, יכול להתווסף לפרויקט Visual Basic או להיות מוצג בדף אינטרנט באמצעות דפדפן. אולם, בתהליך הפיתוח חייבת להיות לך דרך לבחון את פקדי ActiveX ולנפות את השגיאות האפשריות בהם בתוך סביבת הפיתוח של Visual Basic. תהליך זה מורכב מעט יותר מניפוי שגיאות רגיל בפרויקט EXE, מפני שעליך להתמודד עם שני חלקי קוד פעיל: פקד ActiveX עצמו, והפרויקט המשתמש בפקד זה.

בדיקה בעזרת קבוצת פרויקט

הדרך הקלה ביותר לבחון רכיב ActiveX הוא להשתמש בקבוצת פרויקט (**Project Group**). בדוגמאות המוצגות בספר זה, תעבוד עם פרויקט אחד בלבד בזמן נתון, כלומר שכאשר תבחר ביצירת פרויקט חדש מתוך תפריט File של Visual Basic, כל הפרויקטים אשר היו פתוחים ייסגרו. סביבת העיצוב של Visual Basic מאפשרת לך לפתוח כמה פרויקטים בעת ובעונה אחת. סידור זה ידוע כקבוצת פרויקטים והוא שימושי למצב שבו מבקשים לבדוק רכיבי ActiveX. קבוצת פרויקט טיפוסית תכיל את רכיב ActiveX עצמו, ופרויקט בדיקה (הבנוי כפרויקט EXE רגיל).

הוספת פרויקט בדיקה לקבוצה

כדי להתחיל בבדיקת רכיב AddressCtl, עליך להוסיף פרויקט EXE רגיל מתוך תיבת Add Project. חלון סייר הפרויקטים יתעדכן ויציג את שני הפרויקטים, כפי שניתן לראות בתרשים 14.3.

טיפ:

תוכל להוסיף פרויקט רגיל מסוג EXE על ידי לחיצה על הלחצן Add Project.

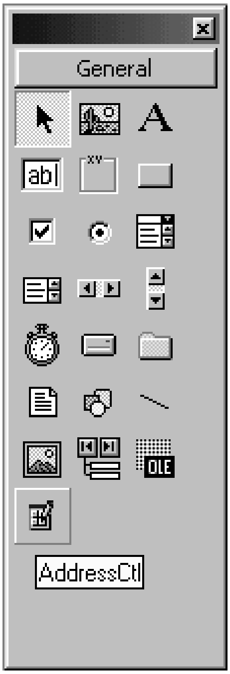


תרשים 14.3: תוכל לפתוח פרויקט נוסף למטרות בדיקת הרכיב

יצירת פרויקט הבדיקה

כאשר הוספת את פרויקט EXE לקבוצה, בוודאי הבחנת שנוסף סמל אפור (מעומעם) לארגז הכלים. סמל זה מייצג את הפקד שלך והוא יישאר מנוטרל כל עוד הטופס UserControl פתוח.

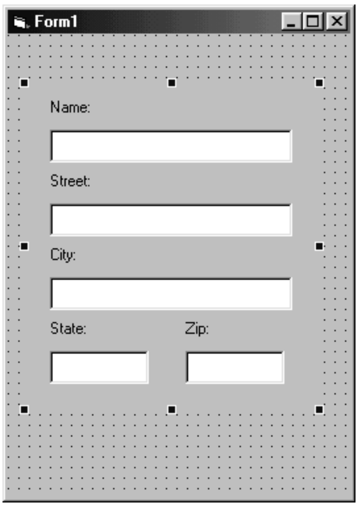
סגור את חלון UserControl ופתח את טופס ברירת המחדל של Project1. הסמל המייצג את הפקד שלך יופיע בארגז הכלים ואמור להיות זמין. כאשר תציג את סמן העכבר מעליו יופיע שם פקד הדוגמה, AddressCtl כמתואר בתרשים 14.4.



תרשים 14.4: כדי להפוך את פקד ActiveX שיצרת לזמין, סגור את חלון Usercontrol

כעת, פקד ActiveX זמין. תוכל לצייר עותק ממנו על הטופס, כמו שאתה עושה עם שאר הפקדים. שים לב שהוא מופיע בדיוק בצורה שבה שרטטת אותו בחלון UserControl, אולם הוא מתנהג כמו פקד בודד. כל תיבות הטקסט בתוך הפקד זזות בתיאום, וכן אין אפשרות לבחור אחת מהן בנפרד מהשאר.

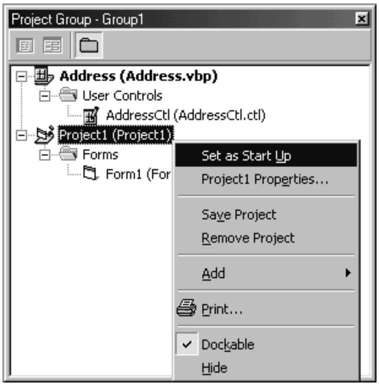
בנקודה זו, אובייקט פקד ActiveX טעון בזיכרון ופועל. נסה לשנות את גודל פקד Address בטופס. כאשר אתה משנה את גודלו בעזרת העכבר, הקוד בשגרת Resize של אובייקט UserControl שומר שגודל תיבות הטקסט השייכות לאובייקט, ישתנה בהתאם. אם תנסה להקטין את גובה הפקד בצורה קיצונית, הפקד יחזור לגודלו המינימלי, כמוצג בתרשים 14.5. מצד שני, תוכל להגדיל את מימדי הפקד ללא הגבלה.



תרשים 14.5: יישום הבדיקה מציג את פקד Address כאשר הוא בגודלו המזערי

יצירת הפרויקט ההתחלתי

כעת, לאחר ששרטטת עותק של הפקד על הטופס, תוכל להתחיל בבדיקת הפרויקט. בקבוצת פרויקטים, עליך להגדיר איזה פרויקט יהיה ה**פרויקט ההתחלתי** (Startup Project). Visual Basic מעבירה מיקוד לפרויקט זה, ומעלה את הפרויקטים האחרים באופן אוטומטי. עליך לקבוע את פרויקט הבדיקה, Project1, כפרויקט ההתחלתי, כדי לבדוק אותו יחד עם פקד ActiveX שיצרת. לחץ לחיצה ימנית על השם Project1 בחלון סייר הפרויקטים, ובחר את האפשרות Set as Start Up מתוך התפריט שיופיע, כמוצג בתרשים 14.6. פרויקט Project1 יודגש כסימול לכך שהוא הפרויקט ההתחלתי.



תרשים 14.6: קבע פרויקט EXE כפרויקט ההתחלתי. פרויקט ActiveX **יופעל** אוטומטית

הפעלת תוכנית הבדיקה

כדי להפעיל את תוכנית הבדיקה, הפעל אותה בעזרת מקש F5 או על ידי לחיצה על הלחצן Start של Visual Basic. הטופס ייטען לזיכרון ותיבות הטקסט של פקד Address תהפוכנה זמינות.

הקלד טקסט כלשהו בכל השדות והקש Ctrl+Break כדי להעביר את הפרויקט למצב שבירה (Break Mode). פתח את חלון Immediate בהקשת Ctrl+G והדפס את ערך מאפיין AddressText על ידי הקלדת הפקודה הבאה:

```
Print AddressCtl1.AddressText
```

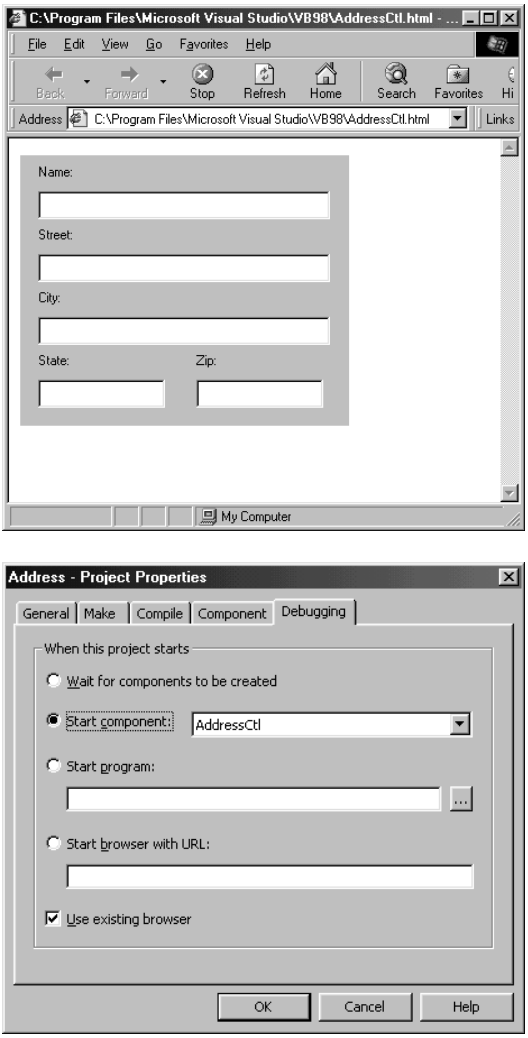
אם תראה את הטקסט שהזנת בתיבות הטקסט, הרי שהפקד עובד כשורה. בנוסף, שים לב שבהקלדת פקודת Print, Visual Basic מציגה את כל המאפיינים השייכים לפקד ב**רשימה מוקפצת** (Pop-Up List). מאפיין AddressText כלול ברשימה.

אם לא תוכל להדפיס את ערך מאפיין AddressText בצורה נכונה, ודא שהקוד בשגרת המאפיין Get הוקלד כראוי. כדי לקבל מידע נוסף על ניפוי שגיאות מתוכניות, ראה פרק 10, ***שליטה במהלך התוכנית***. תוכל להשתמש בשיטות המתוארות כאן כדי לעבור על הקוד בשגרת המאפיין AddressText לבחון את ערכי המשתנים ולוודא תקינותם.

ראה: "ניפוי שגיאות קוד", פרק 10.

בדיקת הפקד בעזרת Internet Explorer

בסעיף הקודם ראית כיצד תוכל להשתמש בפקד Address שיצרת בשילוב עם קבוצת פרויקטים של Visual Basic. כפי שאתה כבר יודע, ניתן להשתמש בפקד ActiveX בצורתו המהודרת בדפדפן Internet Explorer בדף Web. אולם, בגרסאות קודמות של Visual Basic תהליך הבדיקה מעין זה היה קשה. היה עליך ליצור קובץ HTML זמני מחוץ לסביבת הפיתוח של Visual Basic ולהפעיל את Internet Explorer בצורה ידנית. למזלנו, התהליך שופר בגירסה 6.0 של Visual Basic. כעת תוכל להפעיל את Internet Explorer בצורה אוטומטית ולנפות את השגיאות מקוד פקד ActiveX מתוך סביבת העבודה של Visual Basic.



תרשים 14.7: Visual Basic בגירסה 6.0 הופכת את תהליך בדיקת הפקדים למהיר על ידי הפעלה אוטומטית של הדפדפן

אם חלון קבוצת הפרויקטים עדיין פתוח, הסר את פרויקט EXE מהקבוצה. לחץ לחיצה ימנית על Project1 בחלון סייר הפרויקטים ובחר באפשרות Remove Project מתוך התפריט שייפתח. הפרויקט היחיד שאמור להישאר הוא פקד Address.

הפעל את פרויקט הפקד על ידי הקשת F5 או על ידי לחיצה על הלחצן Start של Visual Basic. אם דפדפן Internet Explorer מותקן, הוא אמור לפעול בצורה אוטומטית, כפי שמוצג בתרשים 14.7.

הערה:

בהפעלה ראשונה של הפקד תופיע כרטיסיית Debugging שבתיבת הדו-שיח של מאפייני הפרויקט, כמוצג בתרשים 14.07. כרטיסיה זו קובעת כיצד **יופעלו** רכיבי ActiveX כאשר מקישים F5. (כדי לקבל מידע נוסף בעניין זה, ראה פרק 16, **מחלקות: שימוש חוזר ברכיבים**). לאחר לחיצה על OK **יופעל** הדפדפן Internet Explorer.

לאחר סיום בדיקת הפקד ב-Internet Explorer, סגור את הדפדפן ועצור את הפרויקט.

ראה: "עבודה עם טפסים מרובים", פרק 11.

הידור הפקד

עד כאן, למדת כיצד לבדוק את פרויקט הפקד כאשר הוא מופעל כקוד בסביבת העיצוב. אולם, כדי להשתמש בפקד מחוץ לסביבת העיצוב (כלומר, על מחשב המשתמש), עליך להדר את קוד המקור לפורמט קובץ OCX ולארוז אותו כך שהוא יהיה מוכן להפצה. בצורה זו, יוכלו מפתחים אחרים להתקין את הפקד במחשביהם ולהשתמש בו בפרויקטים שלהם בין אם הם מפתחים בשפת תכנות, מעצבים דף אינטרנט או משתמשים בכלי פיתוח אחר.

יצירת קובץ OCX

תהליך **הידור** (Compiling) פקד ActiveX דומה לתהליך הידור פרויקט EXE רגיל למעט העובדה שהתוצאה הסופית של תהליך הידור פקד ActiveX הנה קובץ OCX ולא קובץ EXE. כדי ליצור קובץ OCX עבור פקד Address, בצע את הפעולות הבאות:

1. בחר באפשרות **Make** מתוך תפריט **File** של Visual Basic.
2. כשתיבת הדו-שיח **Make Project** מופיעה, הזן את שם קובץ למשל ADDRESS.OCX (ייתכן ויהיה עליך לציין גם את שם תיקיה).
3. לחץ על **OK**. Visual Basic תהדר הפקד ותכתוב את תוצאת ההידור בקובץ OCX על הדיסק.

לאחר שפקד ActiveX עבר תהליך הידור, עליך לבדוק את הפקד המהודר. זכור שפקד ActiveX שונה מיישום EXE רגיל בכך שהוא אינו יכול לפעול בצורה עצמאית. כדי לבדוק את הפקד, עליך להשתמש בתוכנית שתכיל אותו.

בדיקת הקוד המהודר

אחרי שהידרת את הפקד, עליו לפעול בדיוק בדרך שבה פועלים פקדים רגילים. התחל פרויקט EXE חדש בסביבת הפיתוח של Visual Basic (הסר את הפרויקטים האחרים הקיימים בזיכרון אם ישנם). לאחר מכן, הצג את תיבת הדו-שיח Components על ידי הקשת Ctrl+T או על ידי בחירת האפשרות Components מתוך תפריט Project. אם הפקד אינו מופיע ברשימה, לחץ על לחצן Browse ובחר את קובץ OCX שלך. כאשר תסגור את תיבת הדו-שיח Components, יווצר סמל חדש בארגז הכלים. במצב זה תוכל לשרטט את הפקד שלך על הטופס כפי שעשית קודם לכן כאשר בדקת את הפקד בקבוצת פרויקטים. הפעם, תוכל להשתמש בקוד מהודר (ראה סעיף קודם **יצירת הפרויקט ההתחלתי**).

הערה:

כאשר אתה משתמש בפקד מהודר ב- Visual Basic, לא תוכל לנפות שגיאות בתוך הפקד עצמו או להציג את קוד התוכנית שלו.

הפצת הפקד למחשב אחר

פקדי ActiveX שלך יכולים לפעול על גבי מחשבי משתמשים ומפתחים אחרים. אולם, תהליך העתקת קובץ OCX סופי למחשב אחר אינו מספיק כדי לגרום לו לפעול. כדי שהפקד יפעל על כל מחשב נתון, אתה חייב לדאוג להתקנת קבצי תמיכה נוספים. הסיבה לדרישה זו היא שהקוד הכתוב בשפת Visual Basic משויך לקבוצת שפות תכנות מדור מתקדם הדורשות קבצי ספריות לשם הפעלתן (קבצי DLL). לדוגמה, אחד הקבצים הבסיסיים הדרושים להרצת כל תוכנית Visual Basic הוא הקובץ MSVBVM60.DLL (Microsoft Visual Basic Virtual Machine 6.0). אף תוכנית Visual Basic אינה יכולה לפעול אם קובץ זה לא יהיה מותקן במחשב.

הערה:

במחשב אשר בו מותקנת Visual Basic (וקבצי התמיכה) ניתן יהיה להשתמש בפקד על ידי העתקת הקובץ למחשב ורישומו במערכת בעזרת פקודת REGSVR32), כבדוגמה הבאה: REGSVR32 C:\WINDOWS\SYSTEM\ADDRESS.OCX.

בשורה התחתונה, כדי שהפקד שלך יוכל לפעול בצורה תקינה, עליו להיות מותקן במערכת. תוכל להתקין אותו בשתי דרכים, בהתחשב בדרך שבה ישתמשו בו:

- **התקנה רגילה** - אם פקד ActiveX שלך מהווה חלק מפרויקט Visual Basic נוסף, הוא יותקן ויירשם במערכת כאשר קבצי היישום יותקנו.
- **התקנת אינטרנט** - אם הפקד שלך מהווה חלק מדף Web, עליך להציב תג <OBJECT> מתאים בקוד HTML. קובץ OCX וקבצי התמיכה, אשר חייבים להיות בתוך ארכיון מסוג CAB על גבי שרת האינטרנט, יועברו ויותקנו על ידי Internet Explorer.

התקנת הפקד בתוכנית התקנה

הדרך הקלה להתקין את הפקד במחשב אחר היא לבצע התקנה רגילה. בהתקנת קובץ EXE רגילה, המשתמש מריץ תוכנית בשם SETUP.EXE אשר דואגת להתקנת הקבצים במחשב ורישומם במערכת. כאשר אתה יוצר קבצי התקנה עבור פרויקט EXE רגיל, השתמש באשף **Package and Deployment**. האשף יידע לטפל בקובץ OCX שלך. אולם, אם אתה חושב שהקובץ שלך ידרוש קבצי תמיכה אשר אינם דרושים לשאר חלקי התוכנית, עליך ליצור קובץ נתמך (Dependency) על ידי האשף. דרך השימוש באשף Package and Deployment מתוארת בנספח 2, **אריזת היישום**.

ראה: "אריזת רכיבי ActiveX", נספח 2.

התקנת הפקד דרך האינטרנט

כפי שידוע לך, יש להתקין את קבצי התמיכה כדי שהפקד יפעל במחשבים אחרים. אם אתה מתכוון להשתמש בפקד ActiveX שלך דרך האינטרנט, Internet Explorer מסוגל להתקין את הפקד בצורה אוטומטית. אולם, צורת העברת הפקד בדרך זו שונה מדרך ההתקנה הרגילה. כאשר המשתמש ניגש לדף Web המכיל פקד ActiveX, Internet Explorer מוריד את קובץ הארכיון (Cabinet-CAB) הדרוש מהשרת ולאחר מכן מתקין את רכיבי ActiveX. זהו אחד מהיתרונות בשימוש בדפדפן Internet Explorer. תוכל להפיץ פקדים מעודכנים על ידי התקנתם בשרת מרכזי אחד. אולם, כדי שהתהליך יפעל כשורה, עליך לספק לדפדפן את כל המידע הדרוש לו.

ראשית, עליך ליצור קבצי CAB עבור פקד ActiveX שלך. קובץ CAB הינו קובץ מסוג מיוחד אשר מכיל קבצים דחוסים. לדוגמה, קובץ CAB בודד יכול להכיל את קובץ OCX המהודר וקבצי הפעלה מסוג DLL של Visual Basic. לא ניתן להשתמש בקבצים הנמצאים בתוך קובץ CAB עד לתהליך חילוצם (Extract) במחשב המשתמש. תוכל להשתמש באשף Package and Deployment כדי ליצור קובץ CAB, כמתואר בנספח 2, **אריזת היישום**. אולם, ייתכן ויהיה עליך לטפל בקבצי CAB בצורה ידנית. ערכת הפיתוח לאינטרנט של Microsoft (Internet Client SDK), אשר ניתנת להשגה דרך אתר Microsoft באינטרנט, מספקת כלים לטיפול בקבצי CAB ממצב MS-DOS.

ניתן גם לשלב חתימה אלקטרונית בקבצי CAB הנוצרים על ידך. תוכל לרכוש קובץ חתימה אלקטרונית, המאפשרת לכל אחד המשתמש בפקד שלך לקבל מידע אודותיך. קובץ זה משמש דרך לאבטחת המידע על מחשב המשתמש מפני שלפקד ActiveX יש גישה מלאה למחשב שעליו הוא פועל. אם לא תשלב קובץ חתימה אלקטרונית בקבצי CAB שתפיץ, משתמש Internet Explorer אשר ירצה להשתמש בפקד שלך יהיה חייב להוריד את הגדרות רמת האבטחה ב- Internet Explorer כדי לאפשר הפעלת פקדים שאינם חתומים בצורה דיגיטלית.

אחת מהתוכניות שבערכת Internet Client SDK, תוכנית בשם SIGNCODE, מאפשרת לשלב חתימה אלקטרונית בקבצים. כאשר אתה מהדר פקד ActiveX לקובץ OCX, המהדר נותן לו שם ייחודי - **Global unique identifier-GUID**. מזהה זה דרוש לתהליך הוספת הפקד לדף Web. כדי לדעת מהו השם הייחודי, תוכל להציץ בקובץ

HTM שנוצר על ידי אשף Package and Deployment. ניתן לראות דוגמה לשימוש בתג <OBJECT> בנספח 2. אחרי שאתה יודע את מזהה GUID של הפקד, תוכל להשתמש בו כדי לשלב את הפקד בדף Web.

ראה: "קבצים המיועדים לשימוש באינטרנט", נספח 2.

השלב האחרון בתהליך הוא להציב את קובץ CAB ואת דף Web (HTML) על השרת. ודא שתג <OBJECT> מכיל את שם קובץ CAB, כמתואר בנספח 3.

אחרי שכל הקבצים הוצבו במקומם, יוכלו משתמשים אחרים להשתמש בפקד. כדי לבדוק שהדף עובד בצורה תקינה, נסה לגלוש אליו מתוך מחשב שהפקד אינו מותקן בו. אם הפקד לא יותקן, בדוק את הדברים הבאים:

- בכרטיסיה **Security**, שבתיבת הדו-שיח **Internet Options** (בתוך תפריט **View**) של **Internet Explorer**, הוסף את שם השרת שלך לרשימת השרתים האמינים. הפעל את הדפדפן מחדש וטען את הדף.
- התקן ורשום את הפקד בעזרת תוכנית התקנה רגילה. בדרך זו, תוכל לוודא שהבעיה קיימת בקובץ הארכיון (CAB).

כאשר אתה פותר בעיות הקשורות בפקדים בדפדפן Internet Explorer, כדאי שתמחק את הקבצים והאובייקטים הנמצאים בזיכרון המטמון (Cache) כדי שתהיה בטוח שאתה מקבל עותק "עדכני" בטעינה חדשה של הדף. תוכל למחוק את הקבצים הזמניים על ידי לחיצה על לחצן **Delete Files** שבכרטיסיה **General** בתיבת הדו-שיח **Internet Options**. אז תוכל להסיר את הפקד על ידי סימונו בתיקיה **\Windows\Downloaded Program Files** ובחירת **Remove** מתוך תפריט **File**.

שיפור פקד ActiveX

הסעיף הקודם הציג את רעיון יצירת פקד ActiveX על ידי ארגון פקדים קיימים בתוך אובייקט UserControl. תוכל ליצור גם פקדים "חדשים" על ידי הוספת אפשרויות לפקדים קיימים, כלומר, תוכל להשתמש בפקד מסוג מסוים ולהוסיף לו שגרות, שיטות ואירועים. לדוגמה, תוכל ליצור פקד פס גלילה (ScrollBar) מיוחד אשר מציג אותיות במקום במספרים, או ליצור תיבת טקסט שתקבל רק סימנים מסוימים.

הצבת הקוד אשר יבצע את הפעולות הללו בפקדי ActiveX נפרדים גורמת לכך שיהיה קל יותר להשתמש בקוד בתוכניות עתידיות. לדוגמה, במקום להוסיף קוד מיוחד לכל פקד תיבת טקסט בתוכנית שלך, תוכל להשתמש בפקד ה"משופר" שתיצור במקום בפקד תיבת טקסט רגיל. כדי ליצור את הפקדים המשופרים הללו, עליך להשתמש באותן הטכניקות שלמדת בסעיפים הקודמים:

1. התחל פרויקט חדש מסוג פקד **ActiveX**.
2. הוסף את הפקד הבסיסי לחלון **UserControl**.
3. הוסף קוד עבור המאפיינים, השיטות והאירועים.

הסעיפים הבאים יוליכו אותך דרך שלבים אלה, תוך שימוש בפקד תיבת טקסט כפקד הבסיסי. פקד תיבת הטקסט ה"משופר" שלך יקבל את השם "Limited Character Text Box"). יהיה לו מאפיין מיוחד אשר יאפשר למשתמש לבחור סט תווים שהפקד יסכים לקבל בתיבה. מאפיין נוסף זה, בשם CharAccept, יאפשר למשתמש להגביל את הטקסט המוזן לתוכה לאותיות בלבד, מספרים בלבד, או שניהם יחד.

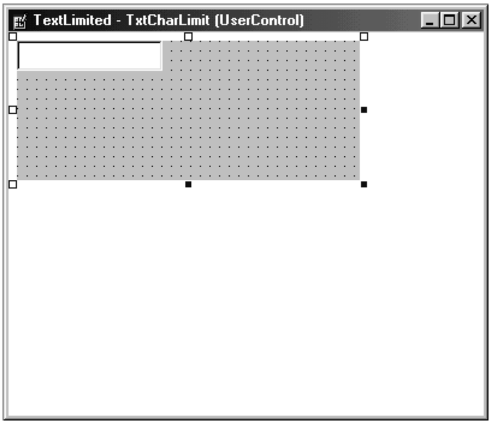
יצירת הפקד הבסיסי

השלבים ביצירת תיבת הטקסט המשופרת דומים לפעולות שביצעת כדי ליצור את פקד Address. כדי ליצור את תיבת הטקסט המשופרת, בצע את השלבים הבאים:

1. התחל פרויקט חדש מסוג פקד **ActiveX**.
2. הוסף תיבת טקסט לחלון UserControl, ומקם אותה כשהפינה השמאלית-עליונה שלה במיקום 0,0.
3. קרא לתיבת הטקסט בשם **txtCharSet** ונקה את מאפיין **Text**.
4. קבע את מאפייני הפרויקט ואת מאפייני אובייקט UserControl לפי טבלה 14.1.

טבלה 14.1: קביעת ערכים עבור תיבת הטקסט המותאמת

שם הפריט	ערך
Project type	ActiveX Control
Project Name	TextLimited
Project Description	Text Box for Limited Character Set
User Control Name property	TxtCharLimit
User Control Public property	True



תרשים 14.8: הפקד הבסיסי, המוצג כאן, ישופר כך שהוא יספק אפשרויות נוספות

כדי לקבוע את שלושת הערכים הראשונים בטבלה, השתמש בתיבת הדו-שיח **Properties**. כדי לקבוע את הערכים האחרים, סמן את אובייקט **UserControl** והקש F4 כדי להציג את חלון המאפיינים. התוצאה הסופית, לאחר שקבעת את ממשק המשתמש של תיבת הטקסט, אמורה להיראות דומה לתרשים 14.8.

5. כתוב קוד בשגרת Resize של אובייקט UserControl כך שגודל תיבת הטקסט יתאים לשטח שהמפתח ישרטט כאשר הוא ישתמש בפקד שלך. שגרת Resize מוצגת בתוכנית הדוגמה 14.1.

תוכנית דוגמה 14.1: LIMITED.VBP - שימוש בשגרת Resize כדי לוודא שהפקד ממלא את כל השטח הריק.

```
Private Sub UserControl_Resize()
	txtCharSet.Height = UserControl.ScaleHeight
	txtCharSet.Width = UserControl.ScaleWidth
End Sub
```

שגרתResize מכילה את כל הקוד הדרוש לממשק המשתמש של פקד הדוגמה. מטרתה לשמור שגודל תיבת הטקסט יהיה זהה לגודל אובייקט UserControl.

6. שמור את הפרויקט.

לפני שתמשיך, בדוק את שגרת Resize על ידי ביצוע הפעולות הבאות:

1. סגור את חלון UserControl. במידה וארגז הכלים גלוי לעין, יופיע עליו סמל חדש המייצג את הפקד שלך.
2. בחר **Add Project** מתוך תפריט **File**. לאחר מכן הוסף פרויקט EXE רגיל.
3. שרטט את הפקד שלך על גבי טופס Form1 ונסה לשנות את גודלו.

המטרה בצעדים אלה היא להרגיל אותך לרעיון שהקוד בפקדי ActiveX שלך פועל גם במצבים שהפקד לא פועל בתוך תוכנית כלשהי. בזמן שאתה מפתח פקד ActiveX, זכור שהקוד שאתה כותב משמש גם בסביבת העיצוב בתוכנית שהפקד ישולב בה.

כעת הסר את הפרויקט מסוג EXE על ידי לחיצה ימנית עליו בחלון סייר הפרויקטים ובחירת האפשרות Remove Project1. עכשיו הגיע הזמן לשפר את הפקד.

שיפור הפקד הבסיסי

השיפורים אשר תבצע בפקד תיבת הטקסט יהיו להגדיר מצבים שבהם הפקד יקבל כל תו מסוג שהוא, ספרות בלבד, או אותיות בלבד. תוכל לעשות זאת על ידי הוספת מאפיין משלך בשם CharAccept, אשר יהיה באחד מהמצבים האפשריים הבאים:

- 0 - הפקד יקבל כל תו.
- 1 - הפקד יקבל ספרות בלבד.
- 2 - הפקד יקבל אותיות בלבד.

יצירת מאפיין CharAccept

כדי ליצור את מאפיין CharAccept, עליך להוסיף משתנה פנימי (Private) אשר יכיל את ערך המאפיין בפקד. לשם כך עליך ליצור אותו באזור ההגדרות הכלליות של אובייקט UserControl בצורה הבאה:

```
Private mCharAccept As Integer
```

לאחר מכן, עליך ליצור את המאפיין החדש הנקרא CharAccept. תוכל ליצור אותו על ידי כתיבת שגרות Get ו-Let בצורה ידנית או על ידי בחירת האפשרות Property מתוך Procedure Add בתפריט Tools. הקוד בשגרת המאפיין CharAccept קל להבנה. כאשר המפתח ירצה לשנות את ערך מאפיין CharAccept, שגרת המאפיין Get תחזיר את ערך המשתנה הפנימי. כאשר ישתנה ערך המאפיין, שגרת Let שלו תשנה את ערך המשתנה הפנימי לאחד מהערכים המותרים. הקוד הבא מיועד לשתי שגרות המאפיין:

```
Public Property Get CharAccept() As Integer
        CharAccept = mCharAccept
End Property
Publick Property Let CharAccept(ByVal nNewValue As Integer)
        Select Case nNewValue
                Case 1 To 2
                        mCharAccept = nNewValue
                Case Else
                        mCharAccept = 0
        End Select
        PropertyChanged "CharAccept"
End Property
```

שים לב לשימוש בשיטת PropertyChanged. שיטה זו עובדת בתיאום עם שגרות ReadProperties ו-WriteProperties, אשר מתוארות בסעיפים הבאים.

כעת עליך להשתמש בשגרת InitProperties של אובייקט UserControl כדי לקבוע למאפיין ערך התחלתי:

```
Private Sub UserControl_InitProperties()
        MCharAccept = 0
End Sub
```

קוד זה מבטיח שלמאפיין יהיה ערך חוקי, אפילו אם המפתח לא דאג לקבוע לו ערך כלשהו.

שימוש באובייקט PropertyBag

עליך ליצור גם את הקוד עבור שגרות ReadProperties ו-WriteProperties כדי לשמור את הגדרות העיצוב של מאפיין CharAccept. שתי שגרות אלו משתמשות באובייקט PropertyBag כדי לשמור ולטעון את ערך מאפיין CharAccept. אובייקט PropertyBag מאפשר לך לטפל בערך הניתן למאפיין בזמן העיצוב. הקוד עבור שתי השגרות, המוצג בתוכנית הדוגמה 14.2, אינו קשה לפענוח. הדבר החשוב הוא לדעת מדוע יש לכתוב את הקוד.

תוכנית דוגמה 14.2: LIMITED.VBP - טיפול בערך המאפיין

```
Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
    m_CharAccept = PropBag.ReadProperty("CharAccept", m_def_CharAccept)
End Sub
Private Sub UserControl_WriteProperties(PropBag As PropertyBag)
    Call PropBag.WriteProperty("CharAccept", m_CharAccept, m_def_CharAccept)
End Sub
```

זכור כי קוד השייך לפקד ActiveX מתחיל לפעול ברגע שאתה משרטט אותו על גבי טופס. נניח שאתה קובע ערך מאפיין כלשהו בסביבת העיצוב. בפקד הדוגמה, הבה נניח שאתה קובע את ערך מאפיין CharAccept ל-1. תוכל לשנות אותו כמה פעמים שתרצה בזמן שהתוכנית פועלת. בדרך כלל, בזמן סיום התוכנית, ערכי הפקד חוזרים למצב שבו הם היו בסביבת העבודה, ולכן יש צורך בטיפול בשני **מצבים** שהמאפיין יכול להיות בהם.

אם ננסה להגדיר זאת בצורה פשוטה יותר, במקרה שאתה משנה ערך מאפיין כלשהו בסביבת העיצוב, יש לדאוג לכך שהפקד ידע להחזיר לעצמו את הערך בזמן פעולת התוכנית. על הפקד גם לדעת לחזור בחזרה לערך זה (אם השתנה) כאשר עוברים בחזרה למצב עיצוב.

אובייקט PropertyBag מאפשר לפקד ActiveX לשמור את ערכי המאפיינים שלו ובכך לענות על הדרישה שהצבנו. שיטת PropertyChanged מאפשרת לנו לדעת שהמשתמש שינה ערך מאפיין כלשהו. כאשר ניתן לדעת באיזה מצב התוכנית נמצאת וכן האם הופעלה שיטת PropertyChanged, Visual Basic תוכל להפעיל את שגרות WriteProperties ו-ReadProperties בזמן הנכון.

הערה:

בוודאי הבחנת בכך שעליך להוסיף כמות רבה של קוד כדי ליצור מאפיין בודד. בסעיף הבא תתוודע לאשף פקדי ActiveX (המגיע כתוספת ל- Visual Basic) אשר הופך את מלאכת הוספת מאפיינים ושגרות אירועים לקלה. אחרי שתסיים ליצור את פקד TxtCharLimit, המשך לקרוא כדי ללמוד כיצד להשתמש באשף זה.

שינוי צורת התנהגות תיבת הטקסט

השלב הבא בפרויקט הדוגמה הוא ליצור את הקוד ההופך את תיבת הטקסט המיוחדת למשהו העושה דבר שונה מפעולת תיבת טקסט רגילה. במקרה זה, תוכל להשתמש בשגרת KeyPress של הפקד כדי לבדוק כל תו כאשר המשתמש מקליד אותו. Visual Basic תעביר את קוד ASCII של התו דרך משתנה KeyAscii של השיגרה. בהתחשב בתו ASCII ובערך מאפיין CharAccept, תוכל לקבל את התו או לשנות את ערך משתנה KeyAscii ל-0, דבר המונע מתיבת הטקסט מלהציג אותו על המסך.

בנוסף לעובדה שהתו לא יוצג על המסך, עליך לעדכן את התוכנית שקראה לשיגרה בעובדה שהמשתמש הקיש תו לא חוקי. תוכל לעשות זאת על ידי יצירת שיגרה בשם UserError. כדי ליצור את השיגרה, הוסף את שורת הקוד הבאה לאזור ההגדרות הכללי של אובייקט UserControl :

```
Public Event UserError()
```

שיגרה זו תתנהג כמו כל שיגרה הקיימת בפקד אחר. כל אחד המשתמש בפקד שלך יכול לכתוב קוד בתוכה. הדבר היחידי שעליך לעשות זה להפעיל את השיגרה על ידי שימוש בשיטת RaiseEvent.

בגלל העובדה שקיימים שלושה סטים אפשריים של תווים, תוכל להשתמש בפקודת Select כדי לטפל באפשרויות השונות. עוד דבר אשר יש לשים לב אליו : עליך לאפשר שימוש במקש Backspace (ASCII code 8) בכל אחד מהסטים המותרים. אחרת, המשתמש לא יוכל למחוק את התו הקודם. הקוד עבור שגרת KeyPress מוצג בתוכנית הדוגמה 14.3.

תוכנית דוגמה 14.3: LIMITED.VBP תכנות שגרת KeyPress להצגת קלט המשתמש.

```
Private Sub txtCharSet_KeyPress(KeyAscii As Integer)
    If KeyAscii = vbKeyBack Then Exit Sub

    Select Case m_CharAccept
        Case 0 'Any character is acceptable
            Exit Sub
        Case 1 'Only number may be entered
            If KeyAscii >= vbKey0 And KeyAscii <= vbKey9 Then
                Exit Sub
            Else
                KeyAscii = 0
                Beep
                RaiseEvent UserError
            End If
```

```
            Case 2 'Only letters may be entered
                If KeyAscii >= vbKeyA And KeyAscii <= vbKeyZ Then
                    Exit Sub
                ElseIf KeyAscii >= 97 And KeyAscii <= 122 Then
                    Exit Sub
                Else
                    KeyAscii = 0
                    Beep
                    RaiseEvent UserError
                End If
    End Select
End Sub
```

הקוד בתוכנית הדוגמה 14.3 פשוט למדי. תקפות KeyAscii, אשר מייצג את התו שהוקלד, נבדקת על ידי פקודות Select Case ו-If. אם התו אינו נמצא בטווח הרצוי, הפקד גורם לצפצוף ומפעיל את שגרת UserError.

בדיקת תיבת הטקסט המגבילה

לאחר שהקלדת את כל הקוד עבור הפקד, תוכל לבדוק את פקד txtCharLimit על ידי ביצוע השלבים הבאים:

1. שמור את הקוד.
2. הוסף פרויקט EXE רגיל לקבוצת הפרויקטים (אם לא עשית זאת עדיין).
3. סגור את חלונות העיצוב והקוד עבור אובייקט UserControl. שים לב לכך שמופיע סמל עבור הפקד שלך בארגז הכלים.
4. הוסף עותק מפקד txtCharLimit לטופס ביישום הבדיקה.
5. קבע את מאפייני הפקד.
6. הפעל את התוכנית ונסה את הפקד. נסה לקבוע את ערך מאפיין CharAccept בערכים שונים כדי לוודא שהוא מקבל רק את התווים הרצויים.

אם אתה נתקל בבעיות בהפעלת הפקד, תוכל להשתמש באותן השיטות לניפוי שגיאות כדי לגלות את הבעיות בפקד כפי שאתה בודק שגיאות בתוכנית רגילה. תוכל לקבוע נקודות עצירה (Breakpoints) ולהפעיל את הקוד שורה אחרי שורה, בין אם בתוך פרויקט פקד ActiveX, או בתוך הפרויקט הרגיל מסוג EXE (ראה פרק 10, **שליטה במהלך התוכנית**).

ראה: "ניפוי שגיאות מתוכניות", פרק 10.

בחירת סמל עבור ארגז הכלים

בוודאי הבחנת שלכל הפקדים שיצרת יש אותו סמל בארגז הכלים. תכונה זו עלולה לגרום לבלבול. תיאור הכלי מספק הסבר עבור הפקד, אך עדיף להגדיר סמלים שונים לפקדים. תוכל להגדיר סמלים על ידי קביעת מאפיין ToolBoxBitmap. מאפיין זה קובע מה יוצג בארגז הכלים. אם הוא נקבע כ-None, הסמל הוא ברירת המחדל. תוכל לקבוע במאפיין כל תמונת Bitmap שהיא, אולם קח בחשבון שגודל סמל ארגז הכלים הוא 15X16 נקודות (Pixels). לפיכך, מומלץ להשתמש בתמונות אשר נוצרו בגודל זה.

אשף ממשק פקדי ActiveX

כאשר יצרת את מאפיין CharAccept לתיבת הטקסט המשופרת, יצרת חלק מממשק הפקד. אך סביר להניח שהמשתמשים ירצו לגשת לרוב המאפיינים, השגרות והאירועים הנפוצים השייכים לתיבת הטקסט. לדוגמה, בוודאי הבחנת שמאפיין Text אינו נגיש. זה הגיוני משום שלא נכתב כל קוד עבור המאפיין. פקד TextBox מהווה אלמנט פנימי בפקד שלך. בשלב הקודם, יצרת את מאפיין CharAccept באופן ידני, אולם כפי שתוכל לשער, דרך זו של יצירת עשרות מאפייני פקד בצורה ידנית מתישה.

למזלנו, Visual Basic מספקת כלי ההופך את התהליך לקל יותר: אשף ממשק פקדי ActiveX - ActiveX Control Interface Wizard. ראשית, עליך להגדיר לאשף את שמות כל המאפיינים שתרצה ליצור עבור הפקד. לאחר מכן הפקד מאפשר לך "לחבר" את מאפייני הפקד שלך למאפייני רכיב הנמצא בתוכו. במילים אחרות, האשף מסוגל לכתוב את קוד השיגרה עבורך ולגרום לכך שמאפיין Text של אובייקט UserControl שלך יעדכן את מאפיין Text של תיבת הטקסט המוכלת בו.

הוספת האשף לסביבת Visual Basic

כדי להשתמש באשף ממשק פקד ActiveX, הוסף אותו קודם לכן לסביבת העיצוב. כדי לעשות זאת, תוכל להשתמש במנהל התוספות (Add-in manager) של Visual Basic. כדי להפעיל את המנהל, בחר באפשרות Add-in Manager מתוך תפריט Add-Ins.

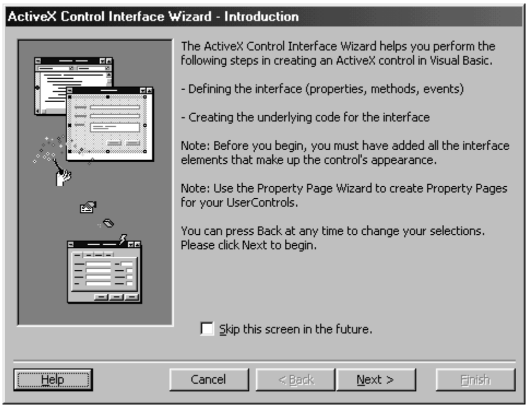
כדי להוסיף את אשף ממשק פקד ActiveX, סמן את תיבת הסימון ליד שם האשף. בכך תודיע למנהל התוספות שברצונך לכלול את האשף בתפריט Add-Ins. לאחר מכן, לחץ על לחצן OK כדי לצאת מנהל התוספות ולהוסיף את האשף לסביבת Visual Basic.

כעת, כדי להדגים את דרך פעולת האשף, צור את תיבת הטקסט המשופרת מחדש:

1. התחל פרויקט חדש מסוג **ActiveX** ושרטט תיבת טקסט בחלון UserControl.
2. קבע את השמות והגדלים באותה הדרך שבה קבעת אותם בדוגמה הקודמת (ראה סעיף **יצירת הפקד הבסיסי**).
3. הפעל את האשף על ידי בחירת **ActiveX Control Interface Wizard** מתפריט **Add-Ins**. האשף יופעל ויציג את המסך ההתחלתי שלו הנראה בתרשים 14.9.
4. לחץ על לחצן **Next** כדי להתחיל את מלאכת יצירת המאפיינים.

טיפ:

כדי שהאשף יפעל בצורה הטובה ביותר, עליך להוסיף את כל הרכיבים הדרושים לפקד שלך לפני שאתה מפעיל אותו.



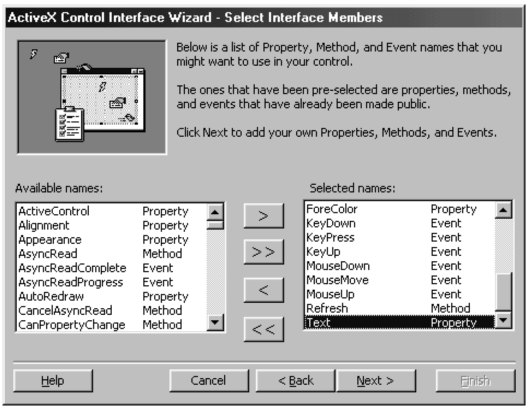
תרשים 14.9: אשף ממשק פקד ActiveX הופך את עבודת יצירת המאפיינים לפשוטה יותר על ידי יצירת חלק גדול מהקוד עבורך

בחירה ויצירה של מאפיינים

השלב הבא בשימוש באשף הוא לבחור את המאפיינים, השיטות והאירועים אשר ברצונך להפוך לזמינים עבור הפקד. באופן כללי, מאפיינים, שיטות ואירועים ידועים בשם **חברים** (Members). בתיבת הדו-שיח Select Interface Members, המוצגת בתרשים 14.10, האשף מכיל את שמות כל החברים שתוכל למצוא בכל פקד בשפת Visual Basic. כדי לבחור מאפיין או שיטה עבור הפקד שלך, סמן את שם החבר ברשימת החברים הזמינים ולחץ על לחצן החץ הימני כדי לבחור את החבר. בפקד הדוגמה, סמן את מאפיין Text בחלון השמאלי, ולחץ על לחצן החץ הימני.

טיפ:

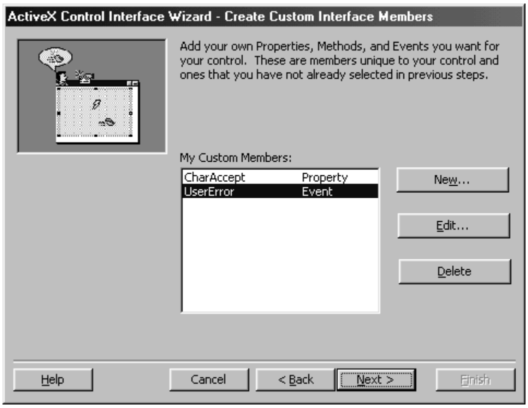
תוכל בנוסף לבחור חבר על ידי לחיצה כפולה על שמו ברשימה.



תרשים 14.10: השלב הראשון ביצירת מאפיינים, שיטות ואירועים עבור הפקד שלך הוא להוסיף שגרות עבור שמות החברים השכיחים

אחרי שהוספת את מאפיין Text לרשימת השמות הנבחרים, לחץ על לחצן Next כדי לעבור לדף הבא של האשף.

לאחר שבחרת את המאפיינים, השיטות, והאירועים עבור הפקד שלך, האשף יעביר אותך לדף שבו תוכל להזין פריטים חדשים המותאמים אישית לפקד. תיבת הדו-שיח Create Custom Interface Members מכילה רשימת כל החברים האישיים אשר ייווצרו עבור הפקד שלך (ראה תרשים 14.11).



תרשים 14.11: האשף מספק גם תיבת דו-שיח אשר בה ניתן להזין שמות פריטים אשר אינם מופיעים בתיבת הדו-שיח בחירת חברי הממשק

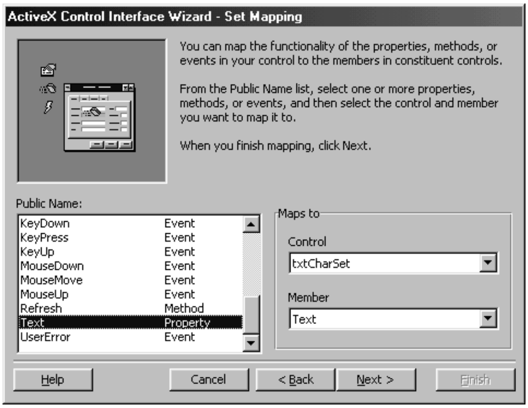
אם יש לך מאפיינים או חברים אחרים אשר הוגדרו מראש, הם יופיעו ברשימה זו כאשר אתה מגיע לתיבת הדו-שיח. מתוך תיבה זו, תוכל להוסיף חברים חדשים ולערוך או להסיר חברים קיימים. כדי להוסיף חבר חדש, לחץ על לחצן **New** של האשף לפתיחת תיבת הדו-שיח הוספת חבר מותאם אישית. בתיבת דו-שיח זו, עליך לקבוע את שם וסוג החבר. צור את מאפיין CharAccept על ידי הקלדת CharAccept בשדה **Name** ולחץ על לחצן OK. אחרי שתחזור לתיבת הוספת חבר מותאם אישית, צור את שגרת UserError באותה הדרך. לבסוף, לחץ על לחצן **Next** כדי להמשיך הלאה.

אזהרה:

אין לערוך או להסיר את חברי הפקד שהגדרת לפני כן בעזרת קוד בלבד מפני שאשף ממשק פקד ActiveX פועל בדרך של ניתוח הערות הממוקמות על ידו בקוד. אין בטחון שהאשף יוכל להבין את הקוד הכתוב על ידך בצורה נכונה.

מיפוי המאפיינים

השלב הבא בשימוש באשף ממשקי ActiveX הוא להקצות את השמות הציבוריים של הפקדים המותאמים אישית לחברים הקבועים בפקד שלך. תהליך זה נקרא **מיפוי החברים** (Mapping the Members). לדוגמה, במקום ליצור מאפיין Text משלך, תוכל למפות את מאפיין Text של הפקד המותאם למאפיין Text של txtCharSet. תיבת הדו-שיח Set Mapping, המוצגת בתרשים 14.12, מכילה רשימה של כל המאפיינים, שיטות והשגרות שהגדרת כחלק מהממשק הציבורי של הפקד שלך.



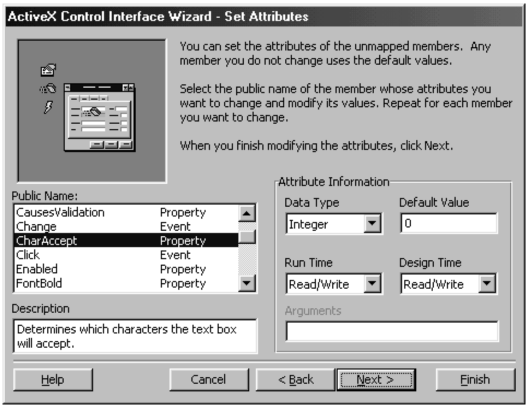
תרשים 14.12: מיפוי החברים הציבוריים של הפקד גורמת לקישורם לחברי הפקדים הנמצאים בפקד

בתיבת הדו-שיח שתי תיבות לזיהוי הפקד וחבר הפקד שאליו החבר הציבורי אמור להיות ממופה. כדי למפות חבר ציבורי יחיד בפקד, בצע את הפעולות הבאות:

1. בחר את החבר ברשימת השמות הציבוריים (Public Names List). בדוגמה זאת, סמן את מאפיין Text בצד השמאלי.
2. בחר את txtCharSet מתוך רשימת הפקדים (רשימה זו מכילה את שמות כל הפקדים הסטנדרטיים בפקד שלך).
3. בחר את החבר השייך לפקד זה מתוך רשימת החברים. במקרה זה, מאפיין Text ייבחר עבורך בצורה אוטומטית. תהליך זה מודגם בתרשים 14.12 עבור מאפיין Text השייך לפקד txtCharLimit.
4. לחץ על לחצן Next כדי להמשיך לתיבת הדו-שיח האחרונה של האשף. תיבה זו, המוצגת בתרשים 14.13, מאפשרת לך לקבוע את ההגדרות עבור כל חבר ציבורי שאינו ממופה לפקד פנימי.

הערה:

בתיבת הדו-שיח Set Mapping, שים לב שתוכל למפות מספר חברים ציבוריים בפעם. יש אפשרות לבחור יותר משם ציבורי אחד. תוכל לבחור כמה שמות ואחר לבחור רכיב שאליו הם ימופו. כל שם ציבורי ימופה למאפיין או השיטה של הרכיב הנושא את אותו שם. לדוגמה, מאפיין Text של הפקד האישי ימופה למאפיין Text של תיבת הטקסט או התיבה המשולבת (Combo Box).



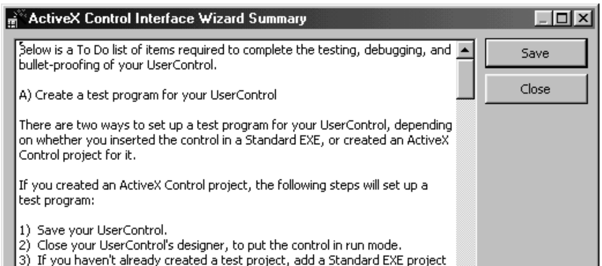
תרשים 14.13: תוכל לקבוע את הגדרות המאפיינים והשיטות לפני השלמת קוד הפקד

האשף מאפשר לך לקבוע הגדרות שונות בהתחשב בסוג החבר שאתה יוצר. עבור מאפיין, תוכל לקבוע את סוג המידע שהמאפיין יוכל להכיל, ערך ברירת המחדל של המאפיין וסוג הגישה המותר למשתמש בסביבת העיצוב ובזמן פעולת התוכנית. סוג הגישה המותר קובע אם תיווצר שגרת המאפיין Let, שגרת המאפיין Get או שניהם יחד עבור המאפיין. תוכל לבחור באפשרויות Read/Write, Read Only, Write Only או None עבור הגישה בזמן פעולת התוכנית. עבור סביבת העיצוב, תוכל לבחור באפשרויות Read/Write, Read Only או None.

עבור פקד הדוגמה, קבע את סוג הנתונים של מאפיין CharAccept כ-Integer ואת ערך ברירת המחדל שלו ל-0. תוכל להוסיף גם תיאור למאפיין בתיבת התיאור. תיאורי מאפיינים מופיעים בתחתית חלון המאפיינים בסביבת העיצוב. כעת, לאחר שהגעת לשלב האחרון של האשף, תוכל ללחוץ על לחצן Finish.

סיום כתיבת הקוד

לאחר שלחצת על Finish, אשף ממשק פקד ActiveX יוצר מספר מודולי קוד בפקד שלך. בנוסף, הפקד גם מציג דף תקציר מידע המספק פרטים אודות השלבים שנשארו כדי לסיים את תהליך יצירת הפקד שלך. דוגמה לדף כזה מוצגת בתרשים 14.14.



תרשים 14.14: קרא את דף התקציר כדי לקבל ייעוץ כיצד לסיים את תהליך יצירת הפקד

אחרי שסקרת את דף התקציר, תוכל להביט בקוד שנוצר על ידי האשף. כדי לראות את הקוד לחץ על אובייקט User Control בחלון הפרויקט. אחר לחץ על View Code.

באזור ההגדרות הכלליות, האשף יצר קבועים לערכי ברירת המחדל של כל המאפיינים שלא מופו. למאפיין CharAccept ניתן ערך ברירת מחדל 0. בנוסף, האשף יצר משתנה פרטי בשם mCharAccept. אחר תופענה ההצהרות על כל שגרת אירוע שבקשת לכלול בפקד. דוגמה מהקוד מוצגת כאן. אם מיפית שגרות אירוע, תראה שהאשף הציב הערות כדי להראות כיצד האירועים ממופים לשגרות אירועים בפקדים שבפקד:

```
Option Explicit
'Default Property Values:
Const m_def_BackColor = 0
Const m_def_ForeColor = 0
Const m_def_CharAccept = 0
'Property Variables:
Dim m_BackColor As Long
Dim m_ForeColor As Long
Dim m_CharAccept As Integer
'Event Declarations:
Event Click() 'MappingInfo=txtCharset,txtCharset, -1,Click
Event DblClick()
Event UserError()
Event KeyPress(KeyAscii As Integer)
```

אחרי הצהרות המשתנים ושגרות האירועים תופענה שגרות המאפיינים. שגרות אלו נוצרות עבור מאפיינים אשר ממופים למאפיין של חבר וכן למאפיינים אשר נוצרו על ידך. כפי שתוכל לראות בקטע הקוד הבא, הקוד עבור מאפיין Text שלם (גם אם פירוש הדבר שקיים שלד שיגרה בלבד עבור מאפיין CharAccept):

```
'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MappingInfo=txtCharset,txtCharset, -1,Text
Public Property Get Text() As String
        Text = txtCharset.Text
End Property
Public Property Let Text(ByVal New_Text As String)
        txtCharset.Text() = New_Text
        PropertyChanged "Text"
End Property
Public Property Get CharAccept() As Integer
        CharAccept = m_CharAccept
End Property
Public Property Let CharAccept(ByVal New_CharAccept As Integer)
        m_CharAccept = New_CharAccept
        PropertyChanged "CharAccept"
End Property
```

אם תרצה לשלב שיטות בפקד, האשף ייצור אותן כפונקציות ולא כתת-שגרות.

לבסוף, הקוד עבור קריאה וכתיבה של ערכי מאפיינים לאובייקט PropertyBag מיוצר בצורה אוטומטית:

```
'Load property values from storage
Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
    m_BackColor = PropBag.ReadProperty("BackColor", m_def_BackColor)
    m_ForeColor = PropBag.ReadProperty("ForeColor", m_def_ForeColor)
    txtCharSet.Text = PropBag.ReadProperty("Text", "")
    m_CharAccept = PropBag.ReadProperty("CharAccept", m_def_CharAccept)
End Sub
'Write property values to storage
Private Sub UserControl_WriteProperties(PropBag As PropertyBag)
    Call PropBag.WriteProperty("BackColor", m_BackColor, m_def_BackColor)
    Call PropBag.WriteProperty("ForeColor", m_ForeColor, m_def_ForeColor)
    Call PropBag.WriteProperty("Text", txtCharSet.Text, "")
    Call PropBag.WriteProperty("CharAccept" m_CharAccept, m_def_CharAccept)
End Sub
```

כדי להשלים את כתיבת הקוד עבור הפקד, עליך להוסיף את הקוד הנחוץ למאפיין CharAccept ולשגרת KeyPress, המוצגת בתוכנית דוגמה 14.4.

תוכנית דוגמה 14.4: LIMITED.VBP השינויים הנחוצים בקוד אשר נוצר על ידי האשף.

```
Public Property Get CharAccept() As Integer
      CharAccept = m_CharAccept
End Property
Public Property Let CharAccept(ByVal nNewValue As Integer)
      Select Case nNewValue
            Case 1 To 2
                  m_CharAccept = nNewValue
            Case Else
                  m_CharAccept = 0
            End Select
            PropertyChanged "CharAccept"
End Property
Private Sub txtCharSet_KeyPress(KeyAscii As Integer)
      If KeyAscii = vbKeyBack Then Exit Sub

      Select Case m_CharAccept
            Case 0 'Any character is acceptible
                  Exit Sub
            Case 1 'Only numbers may be entered
                  If KeyAscii >= vbKey0 And KeyAscii <= vbKey9 Then
                        Exit Sub
```

```
            Else
                KeyAscii = 0
                Beep
                RaiseEvent UserError
            End If
        Case 2 'Only letters may be entered
            If KeyAscii >= vbKeyA And KeyAscii <= vbKeyZ Then
                Exit Sub
            ElseIf KeyAscii >= 97 And KeyAscii <= 122 Then
                Exit Sub
            Else
                KeyAscii = 0
                Beep
                RaiseEvent UserError
            End If
        End Select
End Sub
```

הוספת הקוד הסופי נחוצה מפני שהאשף יוצר "שלד" בלבד עבור חברים אלה. אחרי הכל, אם האשף היה מסוגל ליצור את כל מה שנחוץ, היית מחוסר עבודה!

אשף דפי המאפיינים

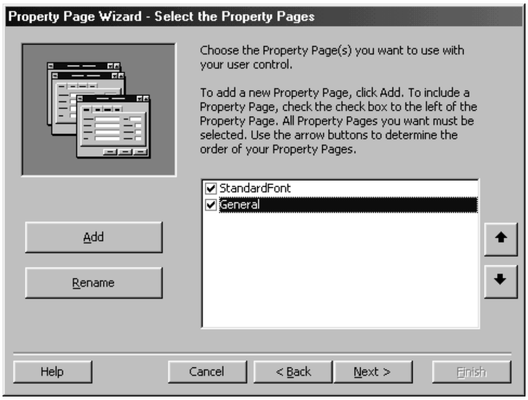
ראית את **אשף דפי המאפיינים** (Property Pages Wizard) בעבר כאשר עבדת עם כמה מהפקדים הפנימיים של Visual Basic. תיבות דו-שיח אלו הופכות את פעולת קביעת המאפיינים השייכים לפקד מסוים לקלה על ידי סידור המאפיינים לפי קבוצות. תוכל ליצור דפי מאפיינים עבור הפקדים שלך תוך שימוש באשף דפי המאפיינים. כמו שעשית באשף ממשק פקד ActiveX, עליך להוסיף את אשף דפי המאפיינים לשולחן העבודה דרך מנהל התוספות. אחרי שביצעת זאת, תוכל להפעיל את האשף מתוך תפריט Add-Ins של Visual Basic. כאשר תפעיל את האשף, תראה מסך המסביר את מטרת השימוש באשף. לחיצה על לחצן Next במסך זה תעביר אותך לתיבת הדו-שיח הראשונה, במקום אשר בו מתחילה מלאכת יצירת דפי המאפיינים.

יצירת הדפים

תיבת הדו-שיח **Select the Property Pages** מאפשרת לך ליצור את דפי המאפיינים (ראה תרשים 14.15). אם כללת את מאפייני Font ו-Color בפקד שלך, האשף ייצור שני דפים בעצמו: StandardColor ו-StandardFont. אם אינך זקוק לדפים אלה, כל שעליך לעשות הוא להסיר את הסימון מהתיבות הנמצאות ליד שמות הדפים כדי למחוק אותם מסט דפי המאפיינים.

בנוסף לדפים המופיעים כברירת מחדל, תוכל להוסיף דפים חדשים לתיבת הדו-שיח. לחיצה על לחצן Add תגרום להופעת תיבת הדו-שיח Property Page Name, אשר בה

תוכל להקליד את שם הדף שברצונך ליצור. כאשר תסיים להזין את השם, הוא יוצב ברשימת הדפים הזמינים ויקבל סימון. סדר שמות הדפים ברשימה הוא הסדר אשר בו תופענה הכרטיסיות. תוכל לשנות את הסדר על ידי סימון דף ושימוש בלחצני החיצים כדי להזיז אותו ברשימה.

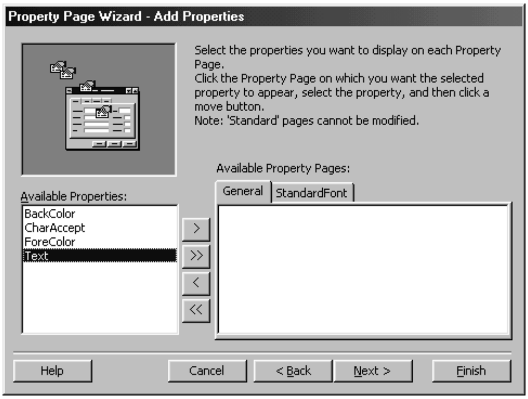


תרשים 14.15: באפשרותך ליצור דפים חדשים או לשנות את שמות הדפים הקיימים באשף דפי המאפיינים

אחרי שסיימת להוסיף דפים לתיבת הדו-שיח, לחץ על לחצן Next כדי לעבור לתיבת הדו-שיח הבאה.

הוספת מאפיינים לדפים

השלב הבא ביצירת דפי המאפיינים הוא להוסיף את המאפיינים הנחוצים לכל דף בתיבת הדו-שיח. תיבת הדו-שיח Add Properties מוצגת בתרשים 14.16.



תרשים 14.16: תיבת הדו-שיח Add Properties מציגה את רשימת המאפיינים הזמינים ומראה את הדפים המוגדרים עבור תיבת הדו-שיח

כדי להוסיף מאפיין לדף כלשהו, לחץ על הכרטיסיה המתאימה לדף הרצוי, בחר את המאפיין מתוך רשימת המאפיינים הזמינים ולחץ על לחצן החץ הימני. שים לב, בתרשים 14.16, לכך שנוסף דף General property ובו מאפיין CharAccept. בנוסף, אם הופיעו דפי ברירת מחדל, StandardColor ו-StandardFont, שים לב לכך שהמאפיינים המתאימים להם כבר נוספו.

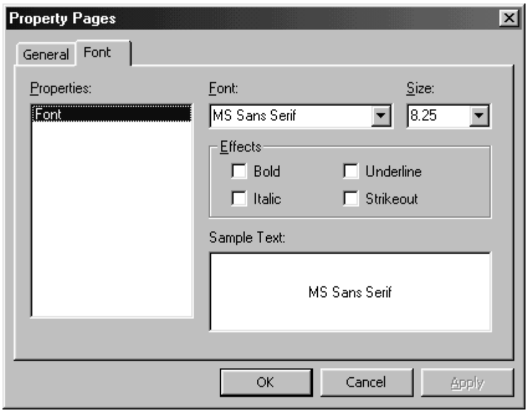
טיפ:

תוכל לגרור מאפיין לכרטיסיה מסוימת כדי להציב אותו בדף המאפיינים הרצוי.

לאחר שסיימת להוסיף מאפיינים לכל הדפים, לחץ על לחצן Finish כדי לסיים את תהליך יצירתם. בדומה לאשף ממשק פקד ActiveX, אשף דפי המאפיינים מציג דף תקציר אשר מספק מידע נוסף שיעזור לך להשלים את יצירת הפקד שלך.

שימוש בדפי המאפיינים ביישומים

כדי להשתמש בדפי המאפיינים שיצרת, עליך להוסיף לפרויקט כלשהו עותק מהפקד שלך. לאחר מכן, בדף המאפיינים, לחץ על הלחצן **בונה** (שעליו שלוש נקודות) הנמצא בשורת המאפיין Custom. כעת תוצג תיבת הדו-שיח דפי המאפיינים כפי שהיא מופיעה בפקדים אחרים ותאפשר למפתח לשנות את ערכי המאפיינים. בתרשים 14.17 מוצגת דוגמה לדף מאפיינים השייך לפקד מותאם אישית.



תרשים 14.17: דף המאפיינים שיצרת עוזר למשתמשים בו להתאים את פקד ActiveX

טיפ:

תוכל גם להגיע לדפי המאפיינים על ידי לחיצה ימנית על הפקד ובחירת האפשרות Properties מתוך התפריט.

יצירה ושרטוט פקד ActiveX

שתי הדוגמאות הקודמות הראו לך כיצד ליצור פקדי ActiveX הבנויים מפקדים קיימים. הסעיפים הבאים ייראו לך איך ליצור פקד אשר נוצר מההתחלה על ידי שרטוט הממשק בעזרת שיטות גרפיות ויצירת המאפיינים, שיטות ואירועים עבור הפקד. הפקד שתיצור בפרק זה יהיה לחצן פקודה בעל הגדרות צבע ורקע שונות, דבר שלא תוכל לעשות תוך שימוש בפקד קיים. למרות העובדה שלחצן פקודה הינו דוגמה פשוטה יחסית, תוכל להשתמש בה כדי להבין כיצד תוכל ליצור פקד משלך.

התחלת הפרויקט

כדי להתחיל ליצור את הלחצן הצבעוני, התחל פרויקט חדש מסוג פקד ActiveX. לאחר מכן, קבע את מאפייני הפרויקט ופקד המשתמש כפי שמתואר בטבלה 14.2.

טבלה 14.2: מאפייני הפרויקט **ו**הפקד

שם הפריט	ערך
Project Name	ColorButton
Project Description	Color Enhanced Command Button
User control Name property	ColorBtn
User control Public property	True

יצירת ממשק המשתמש

בדוגמאות קודמות, שרטטת את ממשק המשתמש של הפקד על גבי אובייקט UserControl על ידי שימוש בפקדים קיימים. אולם, בפקד זה המשורטט על ידך, תוכל ליצור את הממשק בצורה עצמאית בעזרת הוספת קוד לשגרות UserControl. עליך לכתוב את הקוד המשרטט את צורת הפקד בשגרת Paint של אובייקט UserControl. שיגרה זו תופעל בכל פעם שהחבר המכיל את הפקד שלך (כגון טופס התוכנית) משורטט. תוכל גם להפעיל את השיגרה בעצמך על ידי שימוש בשיטת Refresh השייכת לפקד. זכור כי ממשק לחצן הפקודה הינו למעשה תמונה בלבד. עליך להשתמש בשיטות גרפיות, כגון שיטת Line, כדי לשרטט את התמונה שהמשתמש יראה על המסך. לדוגמה, כאשר המשתמש ילחץ על הפקד שיצרת, עליך לדאוג לכך שהתוכנית תשרטט קווים באזור הלחצן בצורה שבה הם ייראו כאילו הלחצן נלחץ. כדי לקבל מידע נוסף בנוגע לשיטות גרפיות, פנה לפרק 19, **שימוש במרכיבי התכנון הוויזואלי**.

ראה: "פקד Line **ו**פקד Shape", פרק 19.

כדי לשרטט לחצן פקודה משולש, תוכל להשתמש בשיטות Line ו-Print בלבד. במקרה זה, הלחצן ימלא את כל שטח UserControl. בתהליך כתיבת הקוד עבור לחצן פקודה צבעוני יש שלושה שלבים:

1. השתמש בשיטת Line השייכת לאובייקט UserControl כדי לשרטט תיבה מרובעת מלאה בגודל המלא של הפקד. כדי ליצור תיבת בגודל המתאים, השתמש במאפייני Height ו-Width של הפקד כמשתני מימדיו בשיטת Line.
2. כדי לדמות את המצב ה"מורם" של הלחצן, שנה את צבעי השורות המרכיבות את התיבה המרובעת. פעולה זו מצריכה שימוש נוסף בשיטת Line כדי ליצור קו לבן לאורך הגבולות עליון ושמאלי של הפקד וכן כדי לשרטט קו שחור לאורך הגבולות תחתון וימני. קווים אלה מעניקים ללחצן את הצורה ה"מורמת" הרגילה שלו (כאשר הלחצן "לחוץ", עליך להחליף את מצבו על ידי החלפת מיקומי הקו הלבן עם הקו השחור).
3. שרטט את תוכן הלחצן על ידי שימוש בשיטת Print.

הקוד אשר יוצר את גוף הלחצן מוצג בתוכנית הדוגמה 14.5. יש להציב את הקוד בשגרת Paint של אובייקט UserControl.

תוכנית דוגמה 14.5: COLORBTN.VBP - שיטת Line לשרטוט הלחצן הצבעוני.

```
Private Sub UserControl_Paint()
        Dim nHeight As Integer
        Dim nWidth As Integer
        With UserControl
                'Leave some room for the border
                nHeight =.Height – 10
                nWidth =.Width – 10
                'Set UserControl's BackColor, draw a colored box
                .DrawWidth = 1
                .FillColor = lngBackColor
                .FillStyle = 0
                UserControl.Line (0, 0) – (nWidth, nHeight),, B
                'Draw lower right lines
                .DrawWidth = 3
                If bMouseDown = False Then
                        .ForeColor = vbBlack
                Else
                        .FireColor = vbWhite
                End If
                UserControl.Line (0, nHeight) – (nWidth, nHeight)
                UserControl.Line (nWidth, 0) – (nWidth, nHeight)
                'Draw upper-left lines
                If bMouseDown = False Then
                        .ForeColor = vbWhite
```

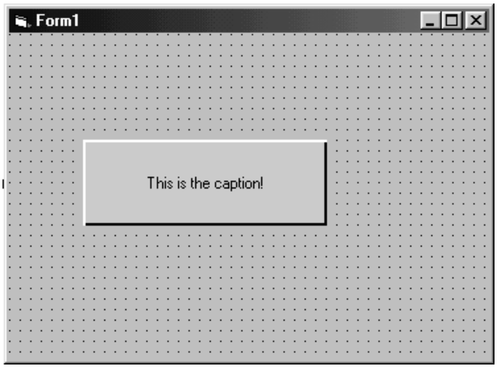
```
                Else
                        .ForeColor = vbBlack
                End If
                UserControl.Line (0, 0) – (nWidth, 0)
                UserControl.Line (0, 0) – (0,nHeight)
                'Calculate and move to the caption position
                .CurrentX = (.Width -.TextWidth(m_Caption)) / 2
                If .CurrentX < 5 Then .CurrentX = 5
                .CurrentY = (.Height – TextHeight(m_Caption)) / 2
                If .CurrentY < 5 Then .CurrentY = 5
                'Draw the Caption
                .ForeColor = lngForeColor
                UserControl.Print m_Caption
        End With
End Sub
```

הקוד בתוכנית הדוגמה 14.5 קל להבנה. ראשית, מתבצעות כמה קריאות לשיטת Line כדי לשרטט תיבה וכמה קווים המייצגים את לחצן הפקודה. לאחר מכן, תוכן הלחצן משורטט במרכז הפקד. שים לב לכך שיש פנייה לכמה משתני מודול בשגרת Paint. לדוגמה, צבע הלחצן נקבע על ידי משתנה lngBackcolor, אשר מכיל את תוכן המאפיין BackColor. יש להצהיר על המשתנים הבאים בסעיף ההגדרות הכלליות של אובייקט UserControl :

```
Dim lngForeColor As Long  'Button Foreground
Dim lngBackColor As Long 'Button Background
Dim bMouseDown As Boolean    'Is the mouse pressed?
```

ההצהרות על ערכי המשתנים הנמצאים ברמת המודול נחוצות לשגרות המאפיינים כדי שהשגרות תוכלנה לשנות אותם. המשתנה הבולאני, bMouseDown קובע אם הלחצן במצב "לחוץ" או "מורם".

בתרשים 14.18 מוצג הצבע הבסיסי של הפקד כפי שהוא נראה בטופס פרויקט הניסיון.



תרשים 14.18: הקוד הקיים בשגרת Paint אחראי לשרטוט ממשק המשתמש של הלחצן

השלב הסופי ביצירת ממשק המשתמש הוא לקבוע ערכים התחלתיים עבור צבעי הלחצן. כדי לעשות זאת, הקלד את הפקודות הבאות:

```
Private Sub UserControl_Initialize()
	LngBackColor = vbCyan
	LngForeColor = vbBlue
End Sub
```

שורות קוד אלו מספקות הגדרות התחלתיות עבור שגרת Paint.

יצירת מאפייני הלחצן

כפי שהזכרנו קודם לכן, תוכל לשמור על עיצוב פשוט של הלחצן. ארבעת המאפיינים העיקריים שלו הם Font, Caption, BackColor ו-ForeColor. שלושה מתוך ארבעת המאפיינים, Font, Backcolor ו-ForeColor, יהיו מקושרים למאפיינים הרלוונטים של פקד המשתמש. בנוסף, המאפיינים BackColor ו-ForeColor יישמרו כמשתנים המיועדים לשימוש על ידי הקוד.

כאשר אתה משתמש באשף ממשק פקד ActiveX, תוכל ליצור את המאפיינים הדרושים לך בקלות. עליך להוסיף את מאפיין Caption ולקבוע אותו כסוג Custom Member וכן למפות את מאפייני Font, BackColor ו-ForeColor לאלה הנמצאים באובייקט UserControl. האשף ייצור את שגרות המאפיינים Let ו-Get עבור ארבעת המאפיינים. קח, לדוגמה, את שגרת המאפיין Get עבור מאפיין BackColor:

```
Public Property Get BackColor() As OLE_COLOR
	BackColor = UserControl.BackColor
End Property
```

שיגרה זו מחזירה את ערך צבע הרקע עבור הפקד. ניתן להשתמש בה כפי שהיא כתובה כאן, אולם עליך להוסיף שורה נוספת לשגרת המאפיין Let, כמתואר בשורות הקוד הבאות:

```
Public Property Let BackColor(ByVal New_BackColor As OLE_COLOR)
	UserControl.BackColor() = New_BackColor
	LngBackColor = UserControl.BackColor
	PropertyChanged "BackColor"
End Property
```

שים לב לכך שנוספה שורה עבור שמירת ערך הצבע במשתנה lngBackColor. שגרת Paint משתמשת במשתנה זה, אשר קיימת עבורו הצהרה באזור ההצהרות הכלליות. למאפיינים הנותרים, המוצגים בתוכנית הדוגמה 14.6, ישנו קוד הכתוב באופן דומה.

תוכנית דוגמה 14.6: COLORBTN.VBP - רוב שגרות המאפיינים נכתבו על ידי האשף.

```
Public Property Get Font() As Font
    Set Font = UserControl.Font
End Property
Public Property Set Font(ByVal New_Font As Font)
    Set UserControl.Font = New_Font
    PropertyChanged "Font"
End Property
Public Property Get ForeColor() As OLE_COLOR
    ForeColor = UserColor.ForeColor
End Property
Public Property Let ForeColor(ByVal New_ForeColor As OLE_COLOR)
    UserControl.ForeColor() = New_ForeColor
    lngForeColor = UserControl.ForeColor
    PropertyChanged "ForeColor"
End Property
Public Property Get Caption As Variant
    Caption = m_Caption
End Property
Public Property Let Caption(ByVal New_Caption As Variant)
    m_Caption = New_Caption
    UserControl_Paint
    PropertyChanged "Caption"
End Property
```

למרות העובדה שעליך להצהיר על המשתנים lngForeGround ו-lngBackGround, אשף ממשק פקד ActiveX יוצר את הגדרת m_Caption עבורך בצורה אוטומטית. הסיבה להבדל נעוצה בעובדה שמאפיין Caption אינו ממופה לאובייקט UserControl. תוכל גם לשנות את ערך הקבוע המכיל את תוכן ברירת המחדל, m_def_Caption למשהו השונה מ-0, כגון השם ColorBtn.

הערה:

האשף **יוצר** גם את הקוד עבור שגרות WriteProperties ו-ReadProperties כדי שניתן יהיה לשמור את ערכי המאפיינים באובייקט PropertyBag.

יצירת אירועי לחצן

תוכל להשתמש באשף ממשק פקד ActiveX גם כדי ליצור את האירועים עבור פקד הלחצן הצבעוני. תוכל לעשות זאת בזמן שאתה יוצר את המאפיינים, או על ידי הפעלה מחדש של האשף. זכור למפות את האירועים Click, GotFocus, MouseDown ו-MouseUp לשגרות המתאימות באובייקט UserControl (ראה סעיף **מיפוי המאפיינים** הנמצא בפרק זה).

לאחר שהאשף יצר את קוד השלד, הוסף את הקוד הנחוץ עבור שגרות MouseDown ו-MouseUp. מן הראוי שהלחצן שתיצור יתנהג בצורה דומה ללחצן פקודה רגיל, ולכן שנה את מראה הלחצן כאשר הוא נמצא במצב "לחוץ". תוכל לעשות זאת על ידי שרטוט קווים שחורים לאורך השוליים העליון והשמאלי של הלחצן כאשר שגרת MouseDown מופעלת וכן על ידי שרטוט קווים לבנים לאורך שוליים אלה כאשר מופעלת שגרת MouseUp. דבר זה נעשה על ידי קביעת ערך המשתנה הבולאני bMouseDown והפעלת הקוד הנמצא בשגרת Paint של אובייקט UserControl. לאחר הוספת הקוד הדרוש לשתי השגרות הללו, הקוד עבור פקד ColorBtn אמור להיראות כפי שמוצג בתוכנית הדוגמה 14.7.

תוכנית דוגמה 14.7: COLORBTN.VBP - קוד השגרות אשר נוצר על ידי האשף ושופר על ידך.

```
Private Sub UserControl_MouseDown(Button As Integer, Shift As Integer, _
        X As Single, Y As Single)
    bMouseDown = True
    UserControl_Paint
    RaiseEvent MouseDown(Button, Shift, X, Y)
End Sub
Private Sub UserControl_MouseUp(Button As Integer, Shift As Integer, X As Single, _
        Y As Single)
    bMouseDown = False
    UserControl_Paint
    RaiseEvent MouseUp(Button, Shift, X, Y)
End Sub
```

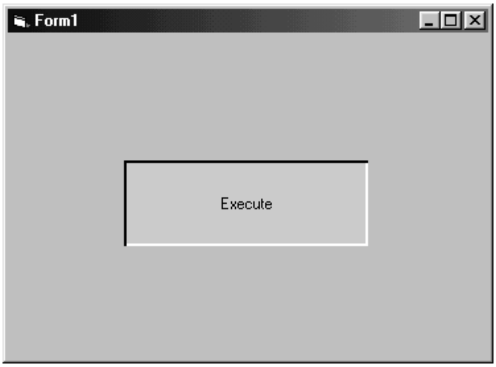
הערה:

אל תערבב את האירועים המתוארים כאן עם אירועי פקד ColorBtn, כפי שהם נראים מנקודת מבט המשתמש. הקוד בתוכנית הדוגמה 14.7 מופעל כאשר מתרחשים אירועים הקשורים בעכבר באובייקט UserControl. הפקודה האחרונה, RaiseEvent, מפעילה את האירוע המתאים בפקד ColorBtn וזה גורם להפעלת קוד בשגרת האירוע אשר נכתב על ידי המשתמש.

תרשים 14.19 מציג את המצב הלחוץ של פקד ColorBtn.

יצירת דפי מאפיינים עבור הלחצן

המשימה האחרונה שעליך לבצע במלאכת עיצוב פקד הלחצן הצבעוני היא ליצור את דפי המאפיינים כדי שהמשתמש יוכל לקבוע את מאפייני הפקד בקלות. השתמש באשף דפי המאפיינים כדי ליצור את הדפים (ראה סעיף **אשף דפי המאפיינים** בפרק זה) ובצע את הפעולות הבאות:



תרשים 14.19: הקווים השחורים מסמלים את העובדה שהלחצן לחוץ

1. הוסף דף **General** לרשימת הדפים.
2. הצב את מאפיין Caption בדף זה.
3. ודא שהמאפיינים BackColor ו-ForeColor נמצאים בדף StandardColor.
4. ודא שמאפיין Font נמצא בדף StandardFont.

בדיקת הלחצן הצבעוני בתוכנית

בנקודה זו, הושלם עיצוב הפקד שלך והוא מוכן לבדיקה בתוך תוכנית. עליך להוסיף פרויקט רגיל מסוג EXE לקבוצת הפרויקטים המכילה את פקד ColorBtn. לאחר מכן עליך לסגור את חלון העיצוב של פקד ColorBtn כדי להפוך את הפקד לזמין עבור פרויקט הבדיקה.

כעת תוכל לשרטט את הפקד על גבי הטופס כפי שאתה משרטט כל פקד אחר. כאשר אתה משרטט את הלחצן, תראה את המסגרת הטיפוסית המסמלת את גודל הפקד. כאשר אתה משחרר את לחצן העכבר, הפקד ישורטט מחדש בצבעי ותוכן ברירת המחדל שלו (ראה תרשים 14.18 כדי לראות כיצד ייראה הפקד).

קביעת מאפייני הלחצן

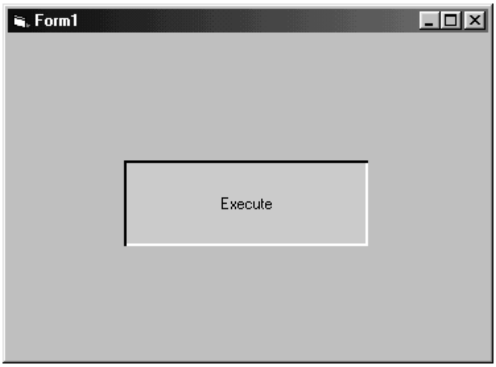
לאחר שהצבת את הלחצן בטופס, קבע את ערכי מאפייניו. תוכל, כמובן, לקבוע את המאפיינים מתוך דפי המאפיינים. נקודה אחת שיש לזכור היא שאם יצרת תיאור במאפיין Caption כאשר הגדרת אותו באשף המאפיינים, הוא יופיע באזור הנמצא מתחת לרשימת המאפיינים כאשר מאפיין Caption נבחר. תיאור זה מהווה עזרה נוספת למשתמש בעבודתו עם הפקד שלך.

דרך נוספת לשנות את מאפייני הפקד שלך היא להשתמש בדפי המאפיינים שיצרת. תיבת הדו-שיח המוצגת בתרשים 14.20 מאפשרת לך לקבוע את המאפיינים Caption, BackColor, ForeColor ו-font השייכים לפקד. שגרת Paint מופעלת בכל פעם שתסגור את דפי המאפיינים. לאחר מכן יוחלו הערכים החדשים על תצוגת הפקד.

כתיבת קוד עבור שגרות האירועים

תהליך כתיבת הקוד עבור שגרת אירוע השייכת לפקד שיצרת זהה לתהליך כתיבת הקוד עבור כל שגרת אירוע אחרת. תוכל ללחוץ לחיצה כפולה על הפקד כדי להגיע לחלון הקוד שלו בפרויקט הבדיקה ולאחר מכן להתחיל להזין את הקוד. כדי לוודא ששגרת Click של הפקד פועלת כשורה, הצב תיבת הודעה (Message box) בשגרת האירוע כדי להודיע שהלחצן נלחץ. הקוד הדרוש לכך מוצג בשורה הבאה:

```
Msgbox "You clicked the ColorBtn Control"
```



תרשים 14.20: דפי המאפיינים מהווים דרך מסודרת אשר בעזרתה יוכלו המשתמשים לקבוע את ערך מאפייני הפקד

לאחר שהוספת את פקודת MsgBox, בדוק את שגרת Click על ידי הפעלת הפרויקט. כאשר תלחץ על לחצן ColorBtn, ההודעה אמורה להופיע. במידה ותרצה לבדוק שגרות אירועים אחרות, בצע את אותם המהלכים.

מכאן...

בפרק זה ראית איך יוצרים פקדי ActiveX, על ידי שיפור פקדים קיימים וכן על ידי יצירתם מחדש. בנוסף, ראית שני אשפים המסופקים על ידי Visual Basic במטרה לעזור לך ביצירת פקדים. כדי לקבל מידע נוסף בנוגע לנושאים אשר נדונו בפרק זה, ראה את המקורות הבאים:

- כדי לקבל מידע נוסף בנוגע ליצירת פקדי ActiveX, ראה פרק 15 **הרחבת פקדי ActiveX**.
- כדי ללמוד על מסמכי ActiveX, ראה פרק 31 **מסמכי ActiveX**.

15

הרחבת פקדי ActiveX

מה בפרק?

- **שימוש באובייקט Ambient לשמירה על אחידות**
- **הצגת האובייקט Extender**
- **בניית פקד Calc**
- **יצירת דפי מאפיינים**
- **טיפול בשגיאות פקדים**

בפרק 14 **יצירת פקדי ActiveX**, ערכת היכרות עם יסודות הבנייה של פרויקט המבוסס על פקדי ActiveX. בנוסף לכך, למדת להשתמש בשני אשפי תוספות (Add In Wizards) אשר סייעו לך בבניית הפקד. פרק זה יעמיק את היכרותך עם פקדי ActiveX, תוך הצגת מספר נושאים נוספים אשר יעזרו לך בבניית פקדים טובים יותר. הסעיפים בפרק זה מכסים את הנושאים הבאים :

- שימוש באובייקטים המשפרים את שילוב הפקדים ב**מכולות** (Containers) שלהם.
- **דפי מאפיינים** (Property Pages) להגברת הגמישות, ללא שימוש באשף.
- טיפול בשגיאות בפקדי ActiveX.

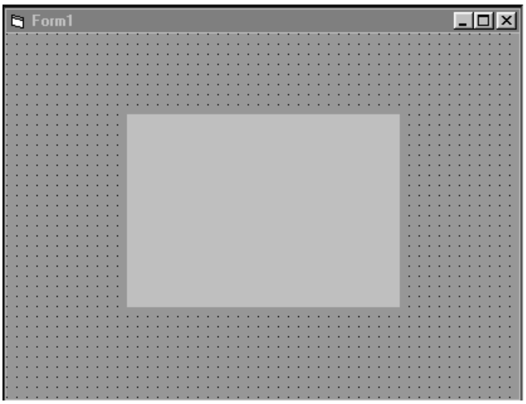
בנוסף לכך, תלמד לבנות פקד ActiveX מעט יותר מורכב : פקד המחשבון.

אובייקט Ambient לשמירה על אחידות

בפרויקט מסוג Standard EXE קיימת מידה רבה של שליטה על מה שהמשתמש רואה, כמו צבע וצורת הטפסים. לעומת זאת, פקדי ActiveX ניתנים למיקום בתוך מכולות שונות, הנקבעות על ידי המשתמש. כדי שהפקד יפעל בצורה עקבית ורצויה בכל סוג מכולה, יש צורך שתהיה לו גישה למידע הנוגע למכולה שלו. למרבה המזל, מפתח מסוגל לכתוב קוד בתוך אובייקט מסוג **פקד משתמש** (UserControl) הנעזר באובייקט הנקרא Ambient (סביבה) לקבל מידע על סביבת הפקד.

יצירת דוגמה לאובייקט Ambient

כדי ללמוד את השימוש באובייקט Ambient, התחל פרויקט חדש מסוג Standard EXE, ואז הוסף פרויקט מסוג ActiveX Control כדי ליצור קבוצת פרויקטים. שנה את גודל אובייקט UserControl כך שיהיה קטן יחסית (בערך 1605 x 1050 יחידות), וסגור את חלון UserControl. אחר שנה את המאפיין BackColor של Form1 לסגול, והצב את הפקד שייצרת על הטופס. התוצאות צריכות להיראות דומות לתרשים 15.1.



תרשים 15.1: פקד משתמש אינו מגיב אוטומטית לשינויי צבע המכולה

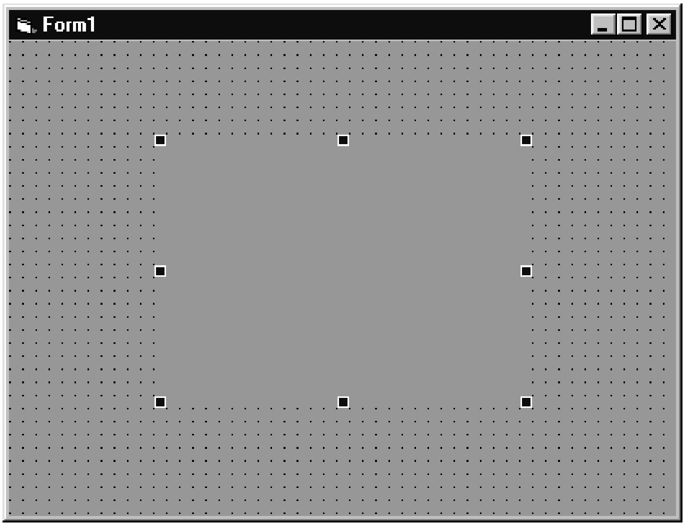
שים לב, כי צבע הרקע של הפקד אינו תואם לצבע הטופס. רוב הסיכויים הם שהפקד יעבוד כראוי, אך הבדלי הצבעים עלולים להסיח את דעתם של המשתמשים. כדי שהפקד יתאים עצמו לרקע אוטומטית, יש למצוא דרך לגרום לקוד הפקד להשיג את מאפיין BackColor של האובייקט המכיל אותו (המכולה). למרבה המזל, האובייקט Ambient מאפשר לפקד לגשת למאפיין זה ולנתונים חשובים נוספים.

מעקב אחר צבעי האובייקט Ambient

כדי להתמודד עם בעיית אי ההתאמה שהוצגה לעיל, יהיה עליך לשנות צבעי הפקד כך שיתאימו לצבעי המכולה שלו. ראשית, מחק את דוגמת הפקד הראשונית שזה עתה יצרת, וחזור לחלון UserControl. כעת, הוסף את שורות הקוד הבאות לשגרת האירוע InitProperties() של UserControl:

```
UserControl.BackColor = Ambient.BackColor
UserControl.ForeColor = Ambient.ForeColor
```

כעת הפקד שלך צריך להתאים עצמו לצבעי המכולה שלו. כדי לוודא זאת, סגור את חלון User Control כך שהפקד יהיה זמין בארגז הכלים. לאחר מכן צייר מופע של פקד המשתמש על הטופס. כאשר הפקד החדש יופיע, עליו להופיע באותו צבע כמו הטופס (ראה תרשים 15.2).



תרשים 15.2: האובייקט Ambient מאפשר להשיג מידע על המכולה של הפקד, כמו למשל, הצבעים

מה יקרה אם משתמשי הפקד ישנו שוב את צבע הרקע? נסה לכוון את מאפיין BackColor לצבע אחר. כשתעשה זאת, הטופס ישתנה מיידית בהתאם לבחירתך, אך הפקד יישאר סגול. תוצאה זו מתקבלת משום שהגדרת קוד באירוע InitProperties, אך לא באירוע החשוב AmbientChanged.

כדי להדגים זאת, הוסף את הקוד הבא לפקד המשתמש:

```
Private Sub UserControl_AmbientChanged(PropertyName as String)
    UserControl.BackColor = Ambient.BackColor
    UserControl.ForeColor = Ambient.ForeColor
End Sub
```

כעת, סגור את חלון User Control, וצייר מופע של הפקד על הטופס. שנה את ערכי המאפיין BackColor, והפעם גם צבעי הרקע של הפקד ישתנו.

מאפייני האובייקט Ambient

כפי שבוודאי ניחשת, האירוע AmbientChanged מופעל כאשר מאפייני האובייקט Ambient משנים את ערכם. הדוגמה לעיל מעדכנת את המאפיינים ForeColor ו-BackColor בכל פעם שהאירוע מופעל, אך הפרמטר PropertyName מאפשר לך לקבוע איזה מאפיין השתנה.

בנוסף למאפייני הצבעים, האובייקט Ambient מכיל כמה נוספים. כמה מהחשובים מהם מופיעים בטבלה 15.1.

טבלה 15.1: מאפיינים חשובים ב-Ambient

מאפיין	תיאור
BackColor	שומר את צבע רקע המכולה
DisplayAsDefault	מציין אם פקד משתמש הינו פקד ברירת המחדל במכולה
DisplayName	שומר את שם מופע הפקד. ניתן להשתמש במאפיין DisplayName כדי לזהות את ההודעות המוצגות בזמן העיצוב
Font	מייצג את גופן המכולה
ForeColor	שומר את צבע הקידמה של המכולה
LocaleID	מציין את המיקום (Locale) בו הפקד רץ. בתוכניות בינלאומיות, נוח להשתמש במאפיין זה כדי לשנות את השפה או תבנית התאריך
TextAlign	מייצג את הגדרת יישור הטקסט של המכולה
UserMode	מציין אם הפקד נמצא בזמן עיצוב או בזמן ריצה. אם ערכו של UserMode הוא False, הפקד נמצא בזמן עיצוב; אחרת הפקד נמצא במצב ריצה. קיים מאפיין נוסף, UIDead, המייצג Break Mode (מצב שבירה)

ניתן לגשת לכל אחד מהמאפיינים בטבלה 15.1 מתוך הפקד על ידי שימוש באובייקט Ambient.

האובייקט Extender

בנוסף למאפיינים המסופקים על ידי האובייקט Ambient, קיים אובייקט נוסף הנקרא **Extender**, המספק גם הוא מספר מאפיינים שימושיים. מאפיינים אלה, הנקראים **Extender properties**, נראים ממבט ראשון כחלק מהתהוות פקד. לדוגמה, כאשר ממקמים פקד על הטופס, המאפיינים Left ו-top נראים כאילו הם חלק מהפקד עצמו. אך בפועל, מאפיינים אלה, כמו גם כמה מאפיינים אחרים, הם חלק מהאובייקט Extender.

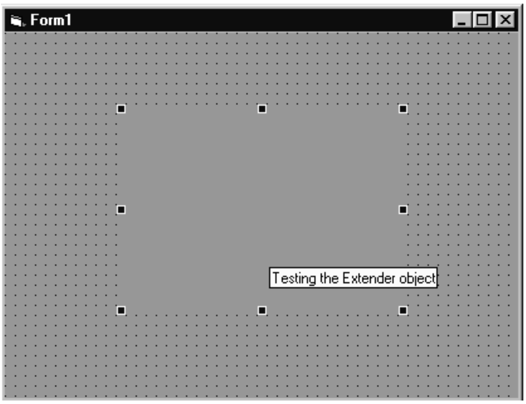
האובייקטים Ambient ו-Extender מספקים מידע על הפקד ביחס למכולה שלו. עם זאת, האובייקט Extender אינו זמין עד אשר הפקד ממוקם על המכולה הלכה למעשה. כיון שכל מכולה שונה מחברתה, לא ניתן לקבוע מראש מה תהיה המכולה עליו ימוקם הפקד או באילו מאפיינים הוא יתמוך. מסיבה זו, לא ניתן לגשת לאובייקט Extender מתוך שגרת האירוע Initialize, הנקראת לפני המיקום הסופי של הפקד במכולה. עם זאת, ניתן לגשת לאובייקט Extender באירועים InitProperties וכן ReadProperties.

כדי לצפות ב- Extender Properties, צייר דוגמה ראשונית של הפקד והקש F4 כדי להציג את חלון המאפיינים. שים לב כי למרות שלא הוספת עדיין מאפיינים לפקד המשתמש, חלק מהם מוצגים ברשימה. אלה הם מאפייני האובייקט Extender.

כדי לבחון את העניין, הוסף את הקוד הבא לאירוע InitProperties של פקד המשתמש:

```
UserControl.Extender.ToolTipText = "Testing the Extender Object"
```

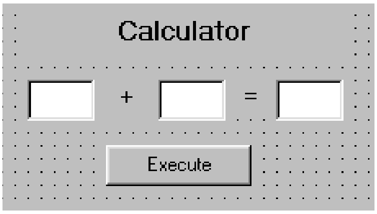
כעת, צייר דוגמה ראשונית של הפקד על הטופס. המאפיין ToolTipText צריך להופיע אוטומטית על דוגמת הפקד שציירת. ניתן לוודא עובדה זו על ידי הצבת מצביע העכבר מעל הפקד, כמוצג בתרשים 15.3.



תרשים 15.3: האובייקט Extender מאפשר לפקד לשנות מאפיינים אשר הוגדרו על ידי המכולה שלו

בניית פקד המחשבון

בסעיף זה תלמד לבנות פקד מחשבון פשוט, אשר ישמש בסיס לנושאים המוצגים בהמשך פרק זה. פקד המחשבון הינו פקד ActiveX פשוט, המאפשר למשתמשים לספק שני ערכי קלט, ולבצע פעולה מתמטית עליהם : חיבור, חיסור, כפל, או חילוק. הפעולה המתמטית נקבעת באמצעות המאפיין Operation, ומבוצעת כאשר המשתמשים לוחצים על לחצן. פקד הדוגמה מוצג בתרשים 15.4.



תרשים 15.4: פקד המחשבון מורכב מ"פקדי משנה" המונחים עליו

יצירת הפקד

כדי ליצור את פקד המחשבון, התחל פרויקט חדש מסוג ActiveX Control. כעת מקם את הפקדים הבסיסיים הבאים על משטח עיצוב פקד המשתמש החדש :

- lblCaption - תווית אשר אליה יכולים לגשת המשתמשים באמצעות המאפיין Caption של הפקד.
- txtNum1 וכן txtNum2 - תיבות טקסט אשר שומרות קלט מהמשתמשים.
- lblOperation - תווית בין תיבות הטקסט, אשר מייצגת את הפעולה החשבונית שתבוצע.
- lblEquals - תווית המציגה את סימן השווה (=) כדי לייצג חזותית את סמל המשוואה המתמטית.
- txtResult - תיבת טקסט המשמשת כדי לשמור את תוצאת החישוב.
- cmdExecute - לחצן המשמש לביצוע הפעולה. המשתמשים יכולים לשנות את כותרת הלחצן באמצעות המאפיין ButtonCaption של הפקד.

לאחר שתסיים למקם את הפקדים, הפקד שלך צריך להיראות כמו הפקד שמוצג בתרשים 15.4.

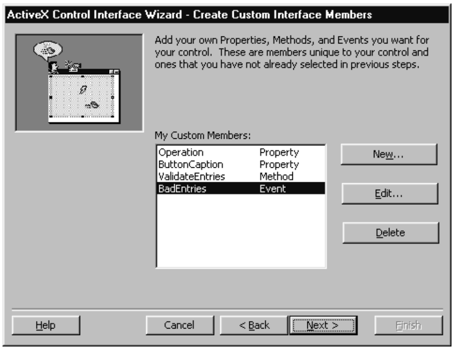
יצירת הממשק

השלב הבא ביצירת פקד המחשבון הינו לכתוב קוד עבור המאפיינים. כזכור לך מהפרק הקודם, אשף ממשק פקד ActiveX מאפשר לך להציג מאפיינים, שיטות ואירועים, השייכים לפקד שלך. דוגמת פקד המחשבון כוללת מספר מאפיינים אותם ניתן להציג בעזרת האשף. בצע את הצעדים הבאים:

ראה: "אשף ממשק פקדי ActiveX", פרק 14

---

1. הפעל את האשף על ידי בחירה באפשרות **ActiveX Control Interface Wizard** מתפריט **Add-ins**. לחץ על **Next**.
2. בתיבת הדו-שיח **Select Interface Members**, הוסף מאפיין **Caption** לפקד.
3. בתיבת הדו-שיח **Create Custom Interface Members**, הוסף שני מאפיינים חדשים, שיטה, ואירוע, כמוצג בתרשים 15.5. מדובר באלמנטים שלהלן:
   - המאפיין **Operation** קובע אם הפקד מבצע פעולת חיבור, חיסור, כפל או חילוק.
   - המאפיין **ButtonCaption** קובע את כותרת הלחצן.
   - השיטה **ValidateEntries** מוודאת את תקפות הקלט בתיבות הטקסט.
   - האירוע **BadEntries** מופעל כאשר הפקד נתקל בקלט שגוי.



תרשים 15.5: אובייקט המחשבון זקוק ל-4 איברי בקרה חדשים

4. בתיבת הדו-שיח **Set Mapping**, מפה את המאפיין **ButtonCaption** למאפיין **Caption** של **cmdExecute**. בנוסף, מפה את המאפיין **Caption** של הפקד למאפיין **Caption** של **lblCaption**.

ביכולתך גם למפות את המאפיינים **BackColor**, **ForeColor** וכן **BorderStyle** למאפיינים המקבילים של אובייקט **UserControl**, אם אתה מעוניין בכך.

5. לחץ על **Finish**, והאשף יצור קוד נוסף לאובייקט **UserContol** שלך.

כעת, כל שעליך לעשות הוא לכתוב את קוד התוכנית למאפיינים, לשיטות ולאירועים.

הגדרת המאפיין Operation

ברירת המחדל של אשף ממשק פקד ActiveX יוצרת את המאפיין Operation כמשתנה מסוג Variant, ומגדירה משתנה פרטי בשם m_Operation כדי לשמור את ערך המאפיין. אך במקרה שלנו, נשתמש בערך שלם (כלומר, משתנה מסוג Integer) כדי לייצג את הפעולה החשבונית: 0 לחיבור, 1 לכפל, 2 לחיסור, ו-3 לחילוק.

כדי שהפקד יהיה נוח לשימוש, מומלץ לא לדרוש מן המשתמש לשנן את משמעות הערכים השרירותיים האלה. לשם כך ניתן להיעזר בהצהרת Enum המאפשרת לפקד להציג את משמעות הערכים במילים. עליך להוסיף את הצהרת Enum לאזור ההצהרות הכלליות של אובייקט User Control, כפי שניתן לראות בתוכנית 15.1.

תוכנית 15.1: CALCTEST.VBG - הצהרת Enum עבור המאפיין Operation.

```
Public Enum OpType
	Add = 0
	Multiply = 1
	Subtract = 2
	Divide = 3
End Enum

Dim m_Operation As OpType ← שנה שורה זו
```

בתוכנית 15.1 נוסף סוג חדש, OpType, ובנוסף סוג המשתנה m_Operation מוחלף מ-Variant ל-OpType. כעת עליך לשנות את השגרות Get ו-Set של המאפיין Operation, כמוצג בתוכנית 15.2.

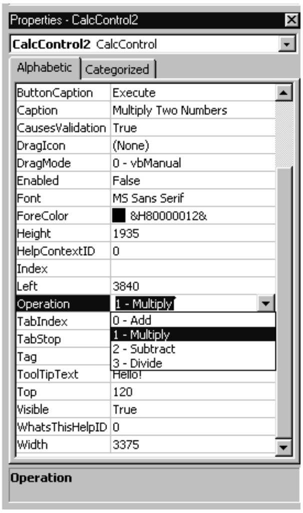
תוכנית 15.2: CALCTEST.VBG - שינוי שגרות המאפיינים לשימוש בסוג OpType.

```
Public Property Get Operation() As OpType
	Operation = m_Operation
End Property

Public Property Let Operation(ByVal New_Operation As OpType)
	m_Operation = New_Operation
	PropertyChanged "Operation"

	Select Case m_Operation
		Case Add
			lblOperation = "+"
		Case Subtract
			lblOperation = "-"
		Case Multiply
			lblOperation = "X"
		Case Divide
			lblOperation = "/"
		End Select
		txtResult = ""
End Property
```

בתוכנית 15.2, בנוסף לשינוי שגרות המאפיינים כך שתשתמשנה בסוג OpType, הוספת קוד המעדכן את lblCaption ומנקה את txtResult כאשר סוג הפעולה החשבונית משתנה. שימוש ב-enum הינו רק דרך אחת לעשות את הפקד שלך יותר נוח לשימוש.
תרשים 15.6 מראה כיצד המאפיין Operation נראה בחלון המאפיינים של הפקד.



תרשים 15.6: המאפיין Operation מספק רשימת ערכים תקפים, במקום לדרוש מהמשתמשים לספק אותם

תכנות שיטות ואירועים

השיטה ValidateEntries קובעת האם המשתמשים הקלידו ערכי טקסט תקפים לשתי תיבות הטקסט. שיטה זו הינה פונקציה פשוטה אשר מחזירה True או False. הקוד לפונקציה זו מוצג בתוכנית 15.3.

תוכנית 15.3: CALCTEST.VBG - האירוע ValidateEntries.

```
Public Function ValidateEntries() As Boolean
	'Assume entries are invalid
	ValidateEntries = False

	'Check empty strings
	if Trim$(txtNum1) = "" Or Trim$(txtNum2) = "" Then Exit Function

	'Check for Numeric values
	If Not IsNumeric(txtNum1) Then Exit Function
	If Not IsNumeric(txtNum2) Then Exit Function

	'If you made it this so far, then they are valid!
	ValidateEntries = True
End Function
```

האירוע Click של הלחצן cmdExecute מכיל את הקוד, אשר מבצע למעשה את החישוב ומציג את התוצאה. כדי ליצור אירוע זה, השתמש בקוד מתוכנית 15.4.

תוכנית 15.4: CALCTEST.VBG - האירוע Click של cmdExecute.

```
Private Sub cmdExecute_Click()
	Dim nVal1 As Integer
	Dim nVal2 As Integer
	Dim nResult As Integer
	If ValidateEntries() = False Then
		RaiseEvent BadEntries
		txtResult = "???"
		Exit Sub
	End If

	nVal1 = Val(txtNum1)
	nVal2 = Val(txtNum2)

	Select Case m_Operation
		Case Add
			nResult = nVal1 + nVal2
		Case Multiply
			nResult = nVal1 * nVal2
		Case Subtract
			nResult = nVal1 - nVal2
		Case Divide
			nResult = nVal1 / nVal2
	End Select

	txtResult = nResult
End Sub
```

בדיקת הפקד

כדי לראות את הפקד מנקודת מבטו של המפתח, סגור את טופס עיצוב הפקד. תוכל ליצור קבוצת פרויקטים ולצרף אליה תוכנית בדיקה מסוג Standard.EXE. עשה זאת באופן הבא: בחר באפשרויות **File**, **Add Project**. פתח את הטופס מתוך פרויקט Standard EXE. צייר דוגמה ראשונית של הפקד על הטופס, והצג את חלון המאפיינים. ביכולתך לכוון את המאפיינים ButtonCaption, Caption, Operation מתוך חלון זה. לחץ לחיצה כפולה על הפקד, והקלד את הקוד באירוע BadEntries:

```
Private Sub CalcControl1_BadEntries()
	MsgBox "You have entered invalid numbers!"
End Sub
```

הרץ את תוכנית הבדיקה, ודא שהמחשבון מפעיל אירוע זה בהקלדת מידע שגוי.

יצירת דפי מאפיינים

פרק 14 הכיל מבוא קצר על **דפי מאפיינים** (Property Pages). דפי מאפיינים הם תיבות דו-שיח המכילות כרטיסיות שנועדו לסייע למשתמש לשנות מאפיינים בזמן עיצוב. כידוע, ניתן להציג דפי מאפיינים באמצעות לחיצה על הלחצן **בונה** (שעליו שלוש נקודות) שבשורת המאפיין **Custom** בחלון המאפיינים, או על ידי לחיצה ימנית על פקד ובחירה באפשרות **Properties**. בנוסף להיותם דרך נוחה לארגון המאפיינים של הפקד, דפי מאפיינים המצורפים לפקד ניתנים לשימוש גם בסביבות פיתוח נוספות, מלבד Visual Basic.

פרק 14 הראה גם כיצד ליצור דפי מאפיינים באמצעות Property Pages Wizard. אשף זה מיועד לאפשר דרך נוחה ליצירת גיליון מאפיינים, אך אין הוא מכסה את כל האפשרויות הקיימות. בסעיף זה תלמד כיצד להכין דפי מאפיינים באופן ידני, כך שתוכל להשתמש בשיטות תכנון ואפשרויות אשר אינן קיימות באשף עצמו.

ראה: "אשף דפי המאפיינים", פרק 14.

להלן השלבים העיקריים בתהליך בניית דפי המאפיינים. הסעיפים הבאים מפרטים כל אחד משלבים אלה.

1. צור אובייקטים של דפי מאפיינים עבור כל קבוצת מאפיינים.
2. מקם על דפי המאפיינים פקדים היכולים לשמש את המפתח לעריכת המאפיינים.
3. יישם את שגרת האירוע SelectionChanged עבור כל דף מאפיינים. אירוע זה אחראי לקריאת הערכים הנוכחיים של מאפייני הפקדים.
4. יישם את שגרת האירוע Change (או לפעמים Click) לכל פקד בכל דף מאפיינים.
5. יישם את שגרת האירוע ApplyChanges בכל דף מאפיינים. מטרתו הינה להעתיק את ערכי המאפיינים החדשים מפקדי דפי המאפיינים אל המאפיינים המקבילים של פקד המשתמש.
6. חבר את דפי המאפיינים אל פקד המשתמש.

יצירת אובייקטים של דפי מאפיינים

הצעד הראשון ביצירת תיבת דפי מאפיינים עבור הפקד, הוא יצירת אובייקטים של דפי מאפיינים לכל קבוצת מאפיינים שברצונך לכלול. הדרך שבה הקבוצות מאורגנות תלויה לגמרי בך, אך רצוי שתשתמש בהגיון בריא. אם לפקד יש מעט מאפיינים, כנראה שתצטרך דף מאפיינים אחד בלבד. אולם, אם פקד כולל מאפיינים רבים, כדאי שתיצור מספר דפים ובכל דף תכלול מאפיינים דומים.

עבור פקד המחשבון לדוגמה, מספיק ליצור דף מאפיינים אחד שייקרא General. בצע את הצעדים הבאים כדי להוסיף את דף זה לפרויקט הפקד.

1. ודא כי פרויקט **ActiveX Control** נבחר בחלון **Project Explorer** ובחר **Project**, **Add Property Page**. תיבת הדו-שיח **Add PropertyPage** מופיעה.

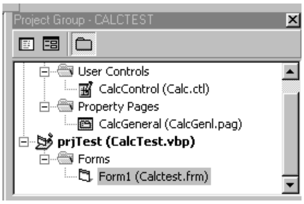
2. לחץ פעמיים על סמל דף המאפיינים, ואובייקט חדש של דף מאפיינים יופיע. שים לב לכך שהוא נראה כמו מעצב טופס.

3. הצג את חלון המאפיינים של דף המאפיינים על ידי הקשה על **F4**.

4. שנה את המאפיין **Name** של דף המאפיינים לערך **CalcGeneral**.

5. שנה את המאפיין **Caption** של דף המאפיינים ל-**General** (כותרת זו מופיעה בכרטיסיה של דף המאפיינים, אשר אינה מוצגת בזמן העיצוב).

דפי המאפיינים החדשים שיצרת אמורים להופיע בחלון Project Explorer, כמוצג בתרשים 15.7.



תרשים 15.7: דפי מאפיינים הינם אובייקטים נפרדים בפרויקט פקד ActiveX

מיקום פקדים על דפי המאפיינים

לאחר שיצרת את דפי המאפיינים, תוכל להוסיף להם את הפקדים הנחוצים כדי לערוך את מאפייני הפקדים המותאמים אישית (Custom Controls), אליהם יקושרו דפי המאפיינים. סוג הפקדים אשר יהיו בשימוש תלוי בך, אך ברוב המקרים תוכל להשתמש בתיבות טקסט כדי לייצג מאפייני טקסט, תיבות סימון כדי לייצג מאפיינים בוליאנים (True/False), תיבות רשימה כדי לייצג מאפיינים בעלי מספר אפשרויות, וכך הלאה. בצע את הצעדים הבאים להוספת פקדים לדפי המאפיינים אשר יצרת:

1. בחלון **Project Explorer**, לחץ פעמיים על אובייקט **CalcGeneral** כדי להציג את הדף **General Property**.

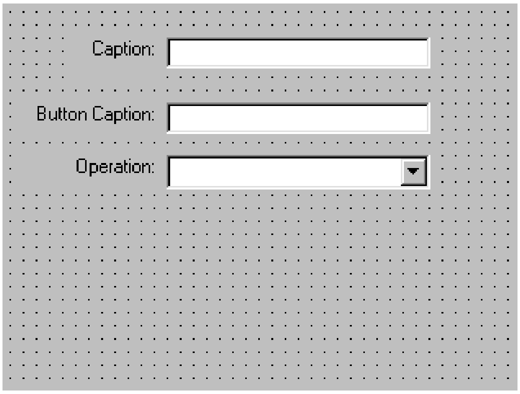
2. הוסף לדף המאפיינים שלושה פקדי **Label**, המכילים את המאפיינים הבאים:

Name	**Caption**
lblCaption	Caption
lblBCaption	Button Caption
lblOperation	Operation

3. הוסף שני פקדי **TextBox**: **txtCaption** ו-**txtButtonCaption**, כדי לשמור את כותרות הפקדים.

4. הוסף תיבה משולבת **cmbOperation**, כדי לשמור את סוג הפעולה החשבונית.

לאחר הוספת פקדים אלה, דף המאפיינים הכללי שלך צריך להיראות דומה לזה המוצג בתרשים 15.8.



תרשים 15.8: דף המאפיינים General עם כל פקדיו

שגרת האירוע SelectionChanged

דמיין לך לרגע שמפתח משתמש בפקד המחשבון כחלק מיישום. המפתח מיקם שני פקדי מחשבון על הטופס, אחד לחיבור והשני לכפל. כאשר המפתח מציג את דפי המאפיינים של אחד או יותר ממפקדי המשתמש, האירוע SelectionChanged של האובייקט Property Object מופעל, כך שכל המאפיינים של פקד המשתמש יוצגו בדף המאפיינים. כדי לכתוב קוד לטעינת המאפיינים אל תיבות הטקסט והתיבות המשולבות, פתח את חלון הקוד של דף מאפייני CalcGeneral על ידי לחיצה כפולה על האובייקט Property Pages. לאחר מכן, הוסף את שורות הקוד מתוכנית 15.5 לשגרת האירוע SelectionChanged.

תוכנית 15.5: CALCTEST.VBG - שגרת האירוע SelectionChanged.

```
Private Sub PropertyPage_SelectionChanged()
	txtCaption.Text = SelectedControls(0).Caption
	txtButtonCaption.Text = SelectedControls(0).ButtonCaption

	cmbOperation.Clear
	cmbOperation.AddItem "0 - Add"
	cmbOperation.AddItem "1 - Multiply"
	cmbOperation.AddItem "2 - Subtract"
	cmbOperation.AddItem "3 - Divide"
	cmbOperation.ListIndex = SelectedControls(0).Operation
	Changed = False
End Sub
```

שים לב כי לכל פקד שעל דף המאפיינים, מוצב ערך ממאפיין של פקד, הנמצא באוסף SelectedControls. כפי שתראה, דפי מאפיינים מסוגלים לשנות מאפיינים של כמה פקדים בו-זמנית. כרגע יצרת דף מאפיינים אשר מסוגל לטפל רק במאפייני פקד יחיד.

שגרת האירוע Change

כאשר המפתח משנה ערך פקד על הטופס, מופעל האירוע Change של הפקד. השתמש באירוע זה כדי ליידע את Visual Basic כי ערך המאפיין בדף המאפיינים שונה.

הערה:

מספר פקדים, כגון CheckBox ו-ComboBox, משתמשים באירוע Click במקום באירוע Change, כדי לטפל בהודעת השינוי.

כל שעליך לעשות הוא לשנות את ערך המאפיין Changed של אובייקט דף המאפיינים לערך True. פעולה זו תאמר ל- Visual Basic להפעיל את הלחצן Apply. הקוד לאירועים Click ו- Change של הדף CalcGeneral מוצג בתוכנית 15.6.

תוכנית 15.6: CALCTEST.VBG אפשור הלחצן Apply באמצעות המאפיין Changed.

```
Private Sub cmbOperation_Click()
      Changed = True
End Sub

Private Sub txtButtonCaption_Change()
      Changed = True
End Sub

Private Sub txtCaption_Change()
      Changed = True
End Sub
```

שגרת האירוע ApplyChanges

האירוע ApplyChanges מבצע פעולה הפוכה מהאירוע SelectionChanged. בעוד ששגרת האירוע SelectionChanged טוענת מידע לדף המאפיינים, שגרת האירוע ApplyChanges שומרת את הערכים החדשים ומחזירה אותם לפקד. אירוע זה מופעל כאשר המפתח סוגר את דף המאפיינים, או לוחץ על לחצן Apply. באשר לפקד המחשבון, עליך פשוט ליטול את המאפיינים מדפי המאפיינים ולהציבם אותם בפקד, כמוצג בתוכנית 15.7.

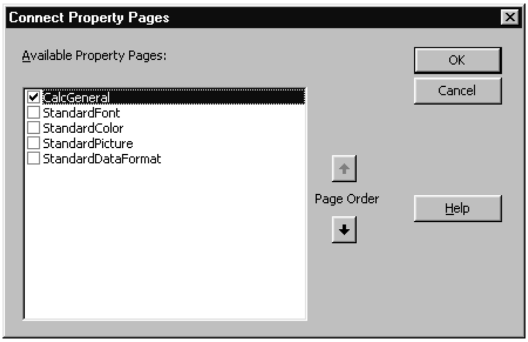
תוכנית 15.7: CALCTEST.VBG - שגרת האירוע ApplyChanges.

```
Private Sub PropertyPage_ApplyChanges()
      SelectedControls(0).Caption = txtCaption
      SelectedControls(0).ButtonCaption = txtButtonCaption
      SelectedControls(0).Operation = cmbOperation.ListIndex
End Sub
```

קישור דף המאפיינים אל הפקד

תמה בניית דף המאפיינים CalcGeneral. עם זאת, כדי שתוכל להשתמש בו בפועל, עליך לקשר אותו לפקד המחשבון (זכור כי דף מאפיינים הינו אובייקט או קובץ נפרד, אשר ניתן לשימוש במספר פקדים). לשם כך, עליך לבצע את הפעולות הבאות:

1. הצג את **Designer Window** של פקד המחשבון.
2. בחלון מאפייני הפקד, לחץ פעמיים על המאפיין **PropertyPages**. תיבת הדו-שיח **Connect Property Pages** מופיעה.
3. סמן את תיבת הסימון **CalcGeneral**, כמוצג בתרשים 15.9.



תרשים 15.9: דפי המאפיינים אשר יסומנו יקושרו אל הפקד

4. לחץ על **OK** בתיבת הדו-שיח כדי לקשר את הדפים הנבחרים עם פקד המחשבון.

שימוש בדף המאפיינים

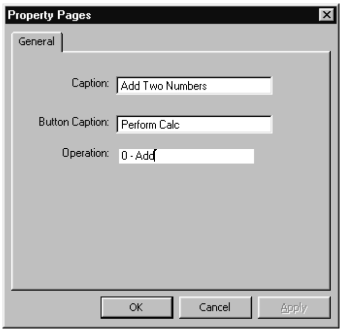
כעת השלמת את דף המאפיינים של פקד המחשבון. כאשר תעבוד עם יישום אשר מכיל פקד זה, תוכל לכוון את מאפייני הפקד מחלון המאפיינים או מדף המאפיינים. כדי לבדוק את דוגמת דף המאפיינים שלנו, ראשית סגור את כל חלונות מעצבי הפקד. כעת, פתח את הטופס מפרויקט הבדיקה, צייר מופע של פקד המחשבון על הטופס, ופתח את חלון המאפיינים שלו על ידי הקשת F4. אם תסתכל מקרוב, תוכל לראות מאפיין חדש הנקרא Custom.

כדי להביט בדף המאפיינים, לחץ לחיצה כפולה על מאפיין Custom, או לחץ על הלחצן **בונה** (שעליו שלוש נקודות). לאחר שתבצע זאת, דף המאפיינים יופיע, כמוצג בתרשים 15.10.

שים לב שמייד לאחר שתשנה פקד על דף המאפיינים, הלחצן Apply יהפוך לזמין.

הערה:

זכור לשמור את השינויים שביצעת בפרויקט המחשבון. שים לב לכך שבזמן שמירת השינויים, דפי המאפיינים ישמרו תחת הסיומת PAG.



תרשים 15.10: הנה דף המאפיינים השלם של פקד המחשבון

בחירת מספר פקדים

כזכור לך מסעיף **שגרת האירוע SelectionChanged**, אוסף האובייקטים SelectedControls מייצג את הפקדים שנבחרו על ידי המפתח בהצגת תיבת הדו-שיח Property Pages. בשגרות האירועים SelectionChanged ו-ApplyChanges ניתן להשתמש באובייקט SelectedControls, כדי לגשת למאפייני כל הפקדים המסומנים. למשל, אם המפתח בחר שני פקדי מחשבון, SelectedControls(0).Caption - המייצג את כותרת הפקד הראשון, ו-SelectedControls(0).Caption - המייצג את כותרת הפקד השני. על ידי כתיבה נכונה של הקוד, המפתח יוכל להחיל את אותה כותרת על שני פקדי המחשבון המסומנים.

ייתכנו מצבים בהם לא תהיה מעוניין לאפשר שינוי מאפיין מסוים, כמו לדוגמה Operation, בכמה פקדים בו-זמנית. במקרים אלה, תוכל להשתמש במאפיין Count של האוסף SelectedControls כדי לבדוק אם יש להתעלם מהשינוי, או אף לשנות את מאפיין Visible של הפקד הנמצא על דף המאפיינים, לערך False.

דרך נוספת לטיפול במקרה מעין זה, היא להחיל שינויים מסוימים על כל הפקדים, ושינויים אחרים רק על הפקד הראשון. אירוע ApplyChanges לדוגמה, אשר מסוגל לטפל במספר פקדים, מוצג בתוכנית 15.8.

תוכנית 15.8: CALCTEST.VBG - החלת שינויים על מספר פקדים.

```
Private Sub PropertyPage_ApplyChanges()
    Dim Control As CalcControl

    'Set Captions for every control
    For Each control In SelectedControls
        control.Caption = txtCaption
        control.ButtonCaption = txtButtonCaption
    Next control

    'Set only the first control's Operation property
    SelectedControls(0).Operation = cmbOperation.ListIndex
End Sub
```

כדי לבדוק שיגרה זו, ראשית פתח את תיבת הדו-שיח Property Pages. לאחר מכן, סמן מספר מופעים של פקד המחשבון, ושנה את מאפייניהם.

טיפול בשגיאות בפקדים

פקדים הפועלים כשורה הינם בעלי חשיבות רבה מאוד. המפתח, כמו גם המשתמש, מסתמכים על הפקד כאובייקט תקין ויציב, אשר אינו אמור להתרסק, אלא בנסיבות יוצאות מגדר הרגיל. המפתח אחראי כמובן, לבדוק כי היישום עובד באופן תקין עם הפקד שלך, אך כדאי גם לך לנקוט מספר אמצעי זהירות כדי למנוע הפתעות לא נעימות למפתחים ולמשתמשים.

ראה: סעיף "אריזת מרכיבי ActiveX", נספח 2.

ראשית, הרבה להשתמש בטיפול בשגיאות כאשר אתה כותב שגרות אירועים. אם מתרחשת שגיאה, והקוד שלך מתעלם ממנה, הפקד יפסיק לפעול ויגרום לנפילת פקד המכולה. ככלל, יש לשלב טיפול בשגיאות בכל האירועים בהם משולב קוד. בנוסף לכך, ישנם מספר תחומי מפתח הזקוקים לתשומת לב מיוחדת:

- **טיפול בקבצים.** מה קורה אם הקובץ אינו קיים או מוגן מפני כתיבה?
- **קישוריות קלט/פלט.** מה קורה אם הרשת נפלה או לא מגיבה, לדוגמה?
- **תקשורת עם פקדים אחרים ועם המכולה.** האם ביכולתך להשתמש בהצלחה בערכי מאפיינים, הנלקחים מפקדים אחרים, בתור קלט לפקד שלך?
- **תנאים גבוליים.** איך הפקד שלך מגיב לערכי קלט מוגזמים או לא נכונים (כמו למשל 99999-)?

בפרק 9 **יסודות התכנות של Visual Basic**, למדת אודות השימוש בהוראה On Error כדי לשלוט על זרימת התוכנית לאחר התרחשות שגיאה.

ראה: "שליטה בשטף התוכנית לאחר שגיאה", פרק 10

הוספת טיפול בשגיאות לשגרה שלך מסתכמת בשורות ספורות של קוד, במיוחד אם הינך דואג ליצור שגרה גלובלית לטיפול בשגיאות, כמו בדוגמה הבאה:

```
Sub ErrorTrap(err as Integer)
    Select Case err
        Case 3021
            sMessage = "Database problem!"
        Case 13
            sMessage = "A System Error has occured"
'... (more error handling code here)
        Case Else
            sMessage = "An unknown Error has occured"
    End Select
End Sub
```

השגרה ErrorTrap יכולה להיקרא מכל אירוע בפקד שלך, באופן הבא:

1. השתמש בהוראה On Error בתחילת השיגרה:

```
On Error Goto Errhandler
```

2. בשיגרה המטפלת בשגיאות, העבר את מספר השגיאה לפונקציה ErrorTrap, בדרך זו:

```
Errhandler:
ErrorTrap err
```

דרך נוספת להימנע משגיאות הינה להשתמש תמיד בשגרות Get ו-Let כדי לגשת למאפייני הפקד, כך שיהיה ביכולתך לוודא את תוקף ערכי המאפיינים בזמן הצבתם. אם אתה יוצר מאפיינים על ידי הצהרת משתנים ציבוריים, תהיה לך שליטה מעטה יותר על ערכיהם. שימוש בשגרות מאפיינים מאפשר לך לעורר שגיאות מיידית אם המשתמשים מנסים להציב ערכים שגויים.

מכאן...

בפרק זה ראית כיצד להעשיר פקדי ActiveX על ידי יצירת דפי מאפיינים. בנוסף לכך למדת על האובייקטים Ambient ו-Extender, ובניית פקד לדוגמה המבצע חישובים. לקבלת מידע נוסף אודות נושאים קשורים, פנה לפרקים הבאים:

- פרק 14 **יצירת פקדי ActiveX**, מספק הבנה בסיסית אודות פקדי ActiveX וכיצד ליצור אותם.
- פרק 31 **מסמכי Activex**, מסביר כיצד להמיר יישום סטנדרטי במהירות לשימוש ב-Web.
- נספח 2 **אריזת היישום**, מספק מידע אודות פריסה של פקדי ActiveX.

16

מחלקות: שימוש חוזר ברכיבים

מה בפרק זה?

- הבנת מחלקות
- בניית מודולים מחלקתיים
- יצירת ActiveX DLL
- יצירת מחלקות המכילות אוספים
- אשף המחלקות

טכניקות תכנות מונחה-אובייקטים (או מונחה עצמים) הינן שימושיות מכיון שהן מאפשרות לבצע ביעילות שימוש חוזר בקוד. בפרק זה תלמד כיצד לבנות מחלקות Visual Basic. **מחלקה** (Class) מאפשרת ליצור אובייקטים, תוך ניצול טכניקות תכנות מונחה-אובייקטים. לדוגמה, בעבודה על פרויקט גדול, כל חבר צוות יכול ליצור מחלקה מסוימת, ולהדר אותה לפורמט של ActiveX DLL, כך שאחרים יוכלו להשתמש בה.

הבנת מחלקות

אם השתמשת בפקדים שהוגדרו על ידי המשתמש ב- Visual Basic, הרי השתמשת כבר במחלקות ואובייקטים. לדוגמה, ציור תיבת טקסט למעשה יוצר **מופע** (Instance) של מחלקת תיבת הטקסט. ציור 5 תיבות טקסט, ייצור 5 מופעים של מחלקת תיבת הטקסט. כל מופע הוא "יישות" בפני עצמו, אך כולם נוצרו מאותה תבנית.

מופעי מחלקות נקראים **אובייקטים** (Objects). כל מחלקה נפרדת היא תבנית, שממנה נוצר סוג מסוים של אובייקט. בדוגמה זו, מחלקת תיבת הטקסט מגדירה כי לתיבת טקסט מאפיין בשם Text. עם זאת, הגדרת המחלקה עצמה אינה מכילה מידע על ערכי המאפיינים. במקום זאת, האובייקט הנוצר מהמחלקה, למשל text1, מכיל את המידע.

תכנות מונחה-אובייקטים

בוודאי הספקת לשמוע את המושג **תכנות מונחה-אובייקטים** (OOP - Object Oriented Programming), או לקרוא עליו. אחד מעקרונות המפתח של OOP הינו הפעלת אובייקטים הניתנים לשימוש חוזר כדי לבנות תוכניות.

OOP מתחיל בשלב התכנון, בקביעת האובייקטים שיופיעו ביישום. לדוגמה, נניח שברצונך לכתוב תוכנית המנהלת תשלומי עובדים. בתוכנית עבודה מסורתית תיקבענה מראש כל הפעולות בתוכנית, כמו "הוספת עובד לבסיס הנתונים" או "הדפסת שיק לעובד". תכנון עבודה מונחה-אובייקטים, ינסה במקום זאת להפריד בין תפקידי התכנות השונים, ובין האובייקטים בתוכנית (עובדים, בסיס הנתונים, שיקים וכדומה). כדי שהתכנון ייחשב מונחה-אובייקטים באמת, כמה עובדות לגבי האובייקטים חייבות להיות נכונות. הרשימה הבאה מכילה את תמצית יסודות OOP:

- **אריזה** (Encapsulation). אריזה, או החבאת מידע, קשורה לכך שהאובייקטים מחביאים את הפרטים על דרך עבודתם. לדוגמה, כאשר אתה משנה את ערך המאפיין Text בתיבת טקסט, אינך יודע (ולא אכפת לך) כיצד תיבת הטקסט מציירת מחדש את התווים. החבאת מידע מאפשרת למפתח האובייקט לשנות את דרך פעולת האובייקט, מבלי להשפיע על המשתמשים באותו אובייקט.

- **ירושה** (Inheritance). ניתן להגדיר אובייקט חדש על בסיס אובייקט קיים, כך שיכיל את כל המאפיינים והשיטות של האובייקט הישן. לדוגמה, ניתן ליצור אובייקט חדש המכיל בנוסף למאפיינים ולשיטות הרגילים של אובייקט קיים, גם כמה משלך. ביכולתך להוסיף מאפיינים נוספים, ו"לרשת" את המאפיינים הקיימים. Visual Basic, במובן הצר של המילה, אינה תומכת בירושה.

- **ריבוי צורות** (Polymorphism). אם כי הרבה אובייקטים יכולים להגדיר שיטות בעלות שם זהה, השיטה יכולה להתנהג בצורה שונה בכל אחד מן האובייקטים. בזכות ריבוי הצורות, התוכנית מריצה את השיטה המתאימה עבור האובייקט הנבחר. למשל, האופרטור + ניתן לשימוש הן עם מחרוזות והן עם מספרים. למרות שאותו סמל (+) נמצא בשימוש בשני סוגי המידע, Visual Basic יודעת לבצע פעולות שונות.

תוצאה חשובה אחת של השימוש ב-OOP הינה קוד הניתן לשימוש חוזר. אחד מהדברים העושים את האובייקט ניתן לשימוש חוזר הוא הממשק, או השיטות והמאפיינים שבהם האובייקט משתמש כדי לתקשר עם שאר העולם. כאשר בונים אובייקטים עם ממשקים מוגדרים היטב, קל יותר לשנות את פנים האובייקט, או אף להוסיף ממשקים חדשים, מבלי להשפיע על תוכניות המשתמשות באובייקט זה.

מחלקות ב- Visual Basic

את המחלקות ב- Visual Basic ניתן ליצור באמצעות **מודול מחלקתי** (Class module). מודולים מחלקתיים יכולים להכיל בתוכם מספר סוגי אלמנטים:

- **מאפיינים:** אלמנטים המשמשים להשמה ואחזור ערכים מהמחלקה.
- **שיטות:** פונקציות או שגרות ציבוריות אשר מוגדרות במחלקה.
- **אירועים:** בדומה לפקדים, אשר מסוגלים לעורר אירועים בטופס אשר מכיל אותם, אובייקט הנוצר ממחלקה מסוגל לעורר אירועים באובייקט המכיל אותו.

בנוסף לאלמנטים אלה, מודולים מחלקתיים מכילים שני אירועים פנימיים: Initialize ו-Terminate. האירוע Initialize מופעל כאשר נוצר מופע חדש של המחלקה, והאירוע Terminate מופעל כאשר האובייקט מושמד.

הגדרות לאובייקטים נוצרות בתוך מודול מחלקתי. מודול מחלקתי הוא כמו מודול קוד רגיל, אך הוא מכיל רק הצהרות משתנים וקוד שגרות. אין בו רכיב ממשק משתמש. אך ניתן להפעיל מחלקה בעזרת טופס הנמצא בתוכנית, ממש כמו מודול קוד רגיל. ניתן להשתמש במודולים מחלקתיים בדרכים שונות, למשל בדרכים הבאות:

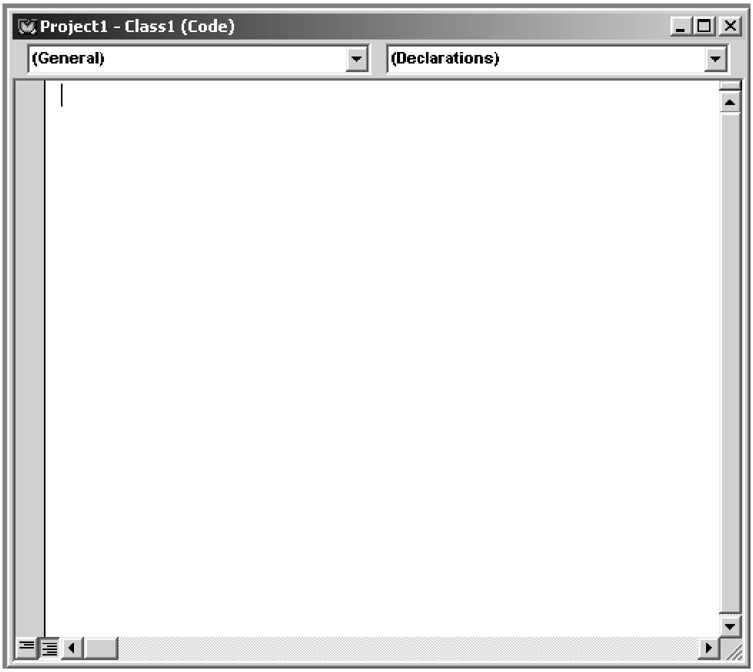
- בפרויקט Visual Basic, מודול מחלקתי מאפשר ליצור מופעים רבים של אובייקטים בכל מקום בתוכניתך, מבלי להשתמש במשתנים גלובליים.
- ביכולתך ליצור אובייקטי ActiveX ולהדר אותם בתור קבצי DLL או EXE, כך שמפתחים אחרים יוכלו להשתמש בהם מתוך הקוד שלהם. לדוגמה, תוכל להגדיר את כל הכללים העסקיים-פיננסיים שלך בתוך מחלקה, ולהדר אותם בתור ActiveX DLL. לאחר מכן, מפתחים אחרים יוכלו לפנות אל אותו DLL ולהשתמש בתוכניתם בכללים הפיננסיים הנמצאים באותו קובץ.
- ביכולתך לבנות תוספת ל- Visual Basic, כדי להעשיר את פונקציות ממשק המשתמש שלה.

בניית מודולים מחלקתיים

הדרך הקלה ביותר ללמוד היא על ידי עשייה, ולכן, הבה ניצור מודול מחלקתי פשוט. מודול מחלקתי זה יכיל מידע אודות העובדים. לאחר שתיצור את המודול, תלמד כיצד להשתמש באובייקט העובדים בתוכניתך.

יצירת מודול מחלקתי חדש

ראשית, צור פרויקט Standard EXE חדש. לאחר מכן, בחר **Add Class Module** מתוך תפריט **Project**. פעולה זו תיצור מודול מחלקתי חדש עם השם ההתחלתי Class1, ותפתח את חלון הקוד של המודול המחלקתי, כמוצג בתרשים 16.1.



תרשים 16.1: על ידי שימוש במודולים ביישומי VB, תוכל ליצור אובייקטים משלך

לאחר יצירת המודול המחלקתי החדש, מומלץ לתת שם ייחודי ותיאורי למחלקה. שם מודול מחלקתי חשוב יותר משם מודול קוד רגיל, משום שאתה משתמש בשם המחלקה בקוד התוכנית. מפתחים רבים נוהגים להוסיף את האותיות c או cls כדי לייצג שם מחלקה. דוגמה פשוטה זו תהיה מחלקה המייצגת פועלים, ולכן נקרא למודול מחלקתי זה clsEmployee.

כדי לתת שם למחלקה, הצג את חלון המאפיינים של המחלקה על ידי הקשת F4 בזמן שחלון הקוד פתוח, או על ידי לחיצה ימנית על המודול המחלקתי בחלון Project Explorer ובחירה באפשרות Properties מהתפריט המקוצר.

הוספת מאפיינים למחלקה

לאחר יצירת המודול המחלקתי, ביכולתך להוסיף לו מאפיינים משלך. המאפיינים במחלקות משמשים לאותה מטרה כמו מאפייני הפקדים: כדי לשמור ולאחזר מידע. ניתן להוסיף מאפיינים למחלקה בשתי דרכים: באמצעות שימוש במשתנים ציבוריים או שגרות מאפיינים.

משתנים ציבוריים

שימוש במשתנים ציבוריים הינה הדרך הפשוטה ביותר ליצירת מאפיינים. פשוט הכנס אותם לסעיף ההצהרות של המודול המחלקתי. לדוגמה של clsEmployee, תוכל להוסיף כמה מאפיינים פשוטים באמצעות הקוד הבא:

```
Public FirstName As String
Public LastName As String
Public Salary As Currency
Public DateHired As Date
```

הערה:

בדרך כלל יהיו במחלקותיך גם מספר משתנים פרטיים, אך רק המשתנים הציבוריים יהיו זמינים לשאר היישום כאשר האובייקט ייווצר.

לאחר שתיצור אובייקט מהמחלקה שלך (מייד תראה כיצד עושים זאת), תוכל להשתמש במאפיינים אלה ממש כמו במאפייני פקד:

```
MyObject.FirstName = "Joe"
MyObject.LastName = "Smith"
MsBox "Hello, " & MyObject.FirstName
```

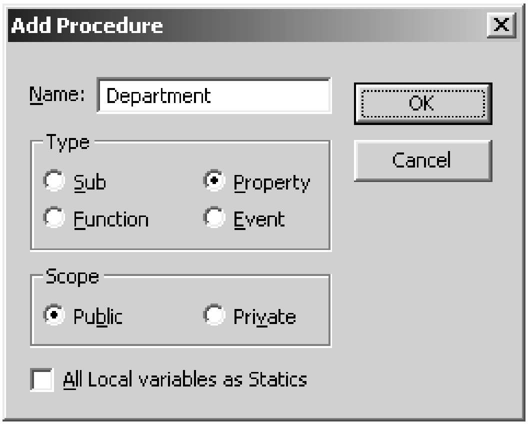
החיסרון בשימוש במשתנים ציבוריים בתור מאפייני אובייקט הוא שלא מבוצע עליהם אימות. בדוגמה זו תוכל להציב כל ערך במאפיינים FirstName, LastName, או DateHired, כל עוד יהיה מהסוג הנכון.

שגרות מאפיינים

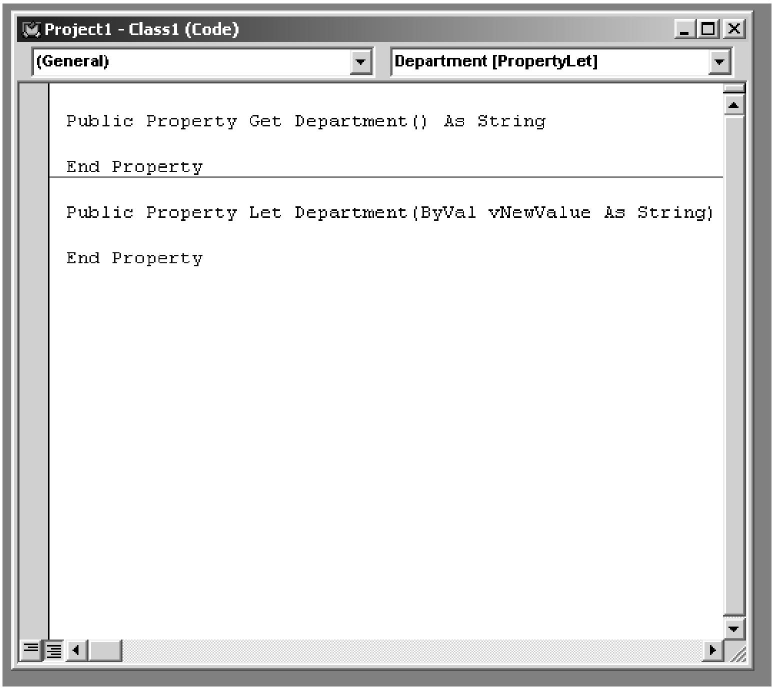
שגרות מאפיינים הינן גמישות יותר מאשר משתנים ציבוריים כיון שהן מאפשרות להריץ קוד כאשר מישהו ניגש למאפיין. שגרות משתנים נכתבות כמו פונקציות, אך מנקודת המבט של משתמשי האובייקט, הן מתנהגות ממש כמו מאפיינים. שלושת הסוגים של שגרות המאפיינים הם כדלקמן:

- Property Get - פונקציה המורצת כאשר המשתמש קורא את ערך המאפיין.
- Property Let - שיגרה המופעלת כאשר המשתמש משנה את ערך המאפיין.
- Property Set - מקרה מיוחד של Property Let, כאשר הערך המועבר לשיגרה הינו בעצמו אובייקט.

שגרות מאפיינים נכתבות בחלון הקוד של המודול המחלקתי. כדי ש- Visual Basic תיצור את השלד של שגרות המאפיינים, הצג את תיבת הדו-שיח **Add Procedure**, המוצגת בתרשים 16.2, על ידי בחירת **Add Procedure** מהתפריט **Tools**. למשל, נניח שתרצה להוסיף מאפיין נוסף למחלקה - המאפיין Department. בתיבת הדו-שיח הקלד **Department** בשדה **Name**, בחר **Property** בתור סוג השיגרה, ולחץ על **OK**.



תרשים 16.2: באפשרותך להשתמש בתיבת הדו-שיח **Add Procedure**, הנגישה מתפריט **Tools**, כדי ליצור סוגים רבים ושונים של שגרות, כולל שגרות מאפיינים



תרשים 16.3: שגרות המאפיינים נראות כמו פונקציות רגילות, אך מנקודת מבט קוד היישום הן מאפייני אובייקט

השימוש בתיבת הדו-שיח Add Procedure גורם ל- Visual Basic להוסיף את תשתית השגרות Property Let ו- Property Get. ברירת המחדל של Visual Basic הינה להניח כי המאפיין הוא מסוג Variant. המאפיין Department יהיה מסוג String, ולכן יהיה צורך לשנות את הגדרות השגרות. שלדי השגרות החדשים מוצגים בתרשים 16.3.

כאשר המשתמשים ניגשים למאפיין Department, שגרת המאפיין המתאימה מופעלת. לרוב, הפעולה הנכונה שיש לבצע היא לשמור את הערך בדרך כלשהי, כמו בדוגמה הבאה, המשתמשת במשתנה פרטי:

```
Public FirstName As String
Public LastName As String
Public DateHired As Date
Public Salary As Currency
Private nDeptNum As Integer

Public Property Get Department() As String
    Select Case nDeptNum
        Case 1
            Department = "Marketing"
        Case 2
            Department = "Accounting"
        Case Else
            Department = ""
    End Select
End Property

Public Property Let Department(ByVal sNewValue As String)
    Select Case Trim$(UCase$(sNewValue))
        Case "MARKETING"
            nDeptNum = 1
        Case "ACCOUNTING"
            nDeptNum = 2
        Case Else
            nDeptNum = 0
    End Select
End Property
```

בקוד למעלה, שגרת המאפיין שומרת למעשה את הערך המייצג את מחלקת העובד בתוך משתנה פרטי של מספר שלם. עם זאת, משתמשי האובייקט תמיד יציבו ויקבלו בחזרה מחרוזת. בתוכנית אמיתית, שגרת המאפיין מסוגלת לבצע פעולות מכל הסוגים על הערך, כולל שמירתו בבסיס נתונים.

הערה:

כדי להגדיר מאפיין לקריאה בלבד, אל תגדיר את השגרות Property Set **או** Property Let **שלו.**

שימוש בפונקציה Property Get עם אובייקטים

מהקוד שבסעיף הקודם מובן שהפונקציה Property Get משמשת להחזרת ערכים מהאובייקט, והשיגרה Property Let משמשת לקבלת ערכים מהתוכנית הקוראת. עם זאת, הפונקציה Property Let אינה פועלת עם אובייקטים. לכן, אם הפרמטר הוא אובייקט, תצטרך להשתמש בשיגרה Property Set, כמו שניתן לראות בקוד הבא:

```
Private objSupervisor As clsEmployee
Public Property Get Supervisor() As clsEmployee
	Set Supervisor = objSupervisor
End Property
Public Property Set Supervisor(objBoss As clsEmployee)
	Set objSupervisor = objBoss
End Property
```

שים לב לפקודה Set, המשמשת לקביעת ערך האובייקט. יש צורך להשתמש בפקודה Set גם בקוד הניגש לשגרות המאפיינים:

```
Dim objEmployee As New clsEmployee
Dim objBoss As New clsEmployee
objEmployee.FirstName = "George"
objEmployee.LastName = "Jetson"
objBoss.FirstName = "Mr."
objBoss.LastName = "Spacely"
Set objEmployee.Supervisor = objBoss

MsgBox "George's Boss is " & objEmployee.Supervisor.LastName
```

הוספת שיטות למחלקה

אובייקטים יכולים להכיל שיטות, כמו שיטת Print (הדפס) של פקד PictureBox. ביכולתך ליצור שיטות במחלקה על ידי הוספת **שגרות ציבוריות** (Public Procedures) למודול המחלקתי. שגרות ציבוריות פועלות כמו שגרות ופונקציות רגילות, אך מילת המפתח Public מצביעה על כך שהן תהיינה זמינות לכל המשתמשים באובייקט שלך. לדוגמה, את הפונקציה הבאה, CalcRaise, ניתן להוסיף לאובייקט העובד:

```
Public Function CalcRaise() As Currency
	Dim sgPercentRaise As Single
	Select Case DateDiff("yyyy", Me.DateHired, Now)
		Case Is > 1
			sgPercentRaise = 0.06
		Case Is <= 1
			sgPercentRaise = 0.02
	End Select
	CalcRaise = Me.Salary * sgPercentRaise + Me.Salary
End Function
```

הפונקציה CalcRaise משתמשת בפרק הזמן השמור במאפיין DateHired כדי לקבוע אם לתת לעובד העלאה של 6 אחוז או של 2 אחוז. הדוגמה מראה כיצד לארוז כלל עסקי במחלקה. פיתוח נוסף של הרעיון יכול להיות למשל הגדרת מספר סוגים נפרדים של מחלקות עובדים (לדוגמה, clsEmployeePartTime ו-clsEmployeeFullTime), כאשר לכל אחד מהם תהיה שיטת CalcRaise נפרדת.

הצהרה ושימוש באובייקטים

לאחר שיצרת את מחלקתך, תוכל ליצור אובייקטים חדשים בתוכנית שלך על ידי פנייה לשם המחלקה:

```
Dim objX As New clsEmployee
```

מילת המפתח New גורמת ליצירת מופע של המחלקה clsEmployee, ולשמירתו במשתנה האובייקט objX. עם זאת, לעיתים תרצה לעכב את יצירת האובייקט לשלב מאוחר יותר בתוכנית. במקרה כזה, השתמש בפרמטר New יחד עם הפקודה Set כאשר אתה מוכן לכך:

```
Dim objX As clsMyClass
'
'Later...
Set objX = New clsMyClass
```

בקוד לעיל, המופע עצמו של האובייקט לא נוצר עד אשר הפקודה Set מופעלת. בדוגמה הראשונה, מופע האובייקט נוצר בפעם הראשונה בה פונים למאפיין או לשיטה השייכים לאובייקט.

הערה:

השתמש תמיד בפקודה Set יחד עם New במקום ליצור את האובייקט עם הפקודה Dim. כך תוכל ביתר קלות לנפות שגיאות מהקוד שלך.

קישור מוקדם לעומת קישור מאוחר

עד עכשיו, כל ההצהרות בפרק זה השתמשו ב**קישור מוקדם** (Early Binding): סוג האובייקט מוגדר בקוד בצורה מפורשת. עם זאת, גם **קישור מאוחר** (Late Binding) הינו אפשרי, כפי שניתן לראות בדוגמה הבאה:

```
Dim objX As Object
Set objX = CreateObject("MyProject.MyClass")
```

כדי לבצע קישור מאוחר, יש להשתמש בפונקציה CreateObject ובמחרוזת מזהה. החסרון של קישור מאוחר הוא: כיוון שהאובייקט objX מוגדר בתור הסוג הכללי Object, לא ניתן לצפות במאפיינים ובשיטות שלו מתוך סורק האובייקטים (Object Browser), ותכונות הכתיבה וההשלמה החכמות לא תהיינה זמינות לגבי אובייקט זה.

עם זאת, לפעמים יש צורך בקישור מאוחר, כיון שהוא מאפשר לשנות את האובייקט מבלי להדר מחדש את התוכנית המשתמשת בו.

הערה:

הפונקציה CreateObject אינה נחוצה כדי להשתמש בקישור מאוחר. ביכולתך לכתוב קוד המשתמש בקישור מאוחר בצורה הבאה:

```
    Dim objX As Object
    Set objX = New MyObj.MyClass
```

דוגמה למקרה בו תהיה מעוניין להשתמש בקישור מאוחר יכולה להיות למשל, בזמן יצירת מופע של אובייקט על מחשב מרוחק.

השמדת האובייקט

הצבת הערך Nothing במשתנה מסוג אובייקט גורמת לשחרור כל המשאבים הקשורים לאובייקט, ולהרצת הקוד הנמצא באירוע הפנימי Terminate. מעשה זה מבצעים באמצעות הפקודה Set:

```
Set objTemp = Nothing
```

טיפ:

למרות שתוכן האובייקטים אינו זמין יותר כאשר המודול או הטופס המכילים אותם מושמדים, הצבת הערך Nothing באובייקטים לאחר סיום השימוש בהם הינה מנהג תכנותי נכון, כיון שכך ניתן לוודא כי כל משאבי הזיכרון משוחררים והאובייקטים מושמדים.

יצירת אובייקטים מפרויקטים אחרים

כאשר ברצונך לגשת למחלקות מפרויקטים אחרים של Visual Basic או מספריות ActiveX DLL, אתה יכול לכלול את שם הפרויקט:

```
Dim objX As New MyProject.clsMyClass
```

זכור כי בשביל לגשת למחלקות בפרויקטים או ספריות DLL, תצטרך להוסיף אותם לרשימת הפניות הפרויקט (Project References).

הוספת אירועים משלך

אובייקטים של Visual Basic מסוגלים לעורר אירועים, כמו האירוע Load הקיים בטפסים. אירועים מספקים דרך נוחה לאובייקט להריץ קוד הנכתב על ידי המשתמש באובייקט. האובייקט מפעיל את האירוע באמצעות **שגרת אירוע** (Event Procedure). למדת כבר כיצד למקם קוד בשגרות אירועים. כדי ליצור אירוע במחלקה שלך, עליך לבצע שני דברים:

1. הצהר על האירוע בתוך המחלקה.
2. השתמש בפקודה RaiseEvent כדי להפעיל את האירוע כאשר נחוץ.

כעת תוכל ליצור אירוע לדוגמה במחלקת העובדים. נניח שתרצה להגדיר דרך להתריע על שגיאות בפני משתמש האובייקט. ראשית, הצהר על אירוע בשם DataError בסעיף ההצהרות הכלליות של clsEmployee, על ידי הוספת שורת קוד בודדת:

```
Public Event DataError(ByVal sMessageText As String)
```

כעת עליך להוסיף קוד כדי להפעיל את האירוע. היזכר בשיטה הקודמת, CalcRaise, המתבססת על המאפיין Salary של האובייקט. אם המשתמש אינו מציב ערך במאפיין זה, לא ניתן לחשב כהלכה את העלאת השכר. הינך יכול לשנות את CalcRaise כדי לשלוח על כך הודעת שגיאה למשתמש דרך האירוע DataError על ידי הוספת פקודת If פשוטה:

```
If Me.Salary = 0 Then RaiseEvent DataError("SALARY UNSPECIFIED!!")
```

כפי שניתן לראות, מאוד קל להוסיף אירוע. כדי לגשת לאירועים שלך מהקוד הקורא, עליך להצהיר על האובייקט באמצעות מילת המפתח WithEvents, ולהגדיר שגרת אירוע אשר ניתנת לקריאה על ידי האובייקט. לדוגמה, מיקום הקוד הבא בחלון ההצהרות הכלליות של טופס או מודול, יגרום ללכידת האירוע DataError:

```
Dim WithEvents objBoss As clsEmployee
Private Sub objBoss_DataError(ByVal sMessageText As String)
    MsgBox sMessageText, vbCritical, "Error!!!"
End Sub
```

מילת המפתח WithEvents גורמת להוספת שם האובייקט לתיבת האובייקטים בחלון הקוד, ומאפשרת לתוכנית Visual Basic ליצור את שלדי שגרות האירועים למענך.

כדי לבדוק דוגמה זו, צור את האירוע DataEvent במודול המחלקתי, והוסף קוד כדי ללכוד את האירוע, כמפורט לעיל. לאחר מכן תוכל לגרום לאובייקט להפעיל את האירוע על ידי קריאה לשיטה CalcRaise כאשר המאפיין Salary שווה לאפס.

יצירת ActiveX DLL

הוספת מחלקות לפרויקטים מסוג Standard EXE הינה שימושית, אך הידורן למרכיבי ActiveX DLL או ActiveX EXE עשוי להיות שימושי יותר. מרכיבי ActiveX אלה ניתנים להתקנה ורישום על מחשב אחר, כך שהיישומים על אותו מחשב מקבלים את היכולת לגשת למחלקות שלך. לדוגמה, תוכל ליצור ספריית DLL המאפשרת גישה לבסיסי נתונים, ולהתקין אותה על שרת Web לשימוש עם עמודי השרת האקטיביים (Active Server Pages). הסעיפים הבאים מתארים כמה מהשלבים הכרוכים ביצירת פרויקט ActiveX.

יצירת פרויקט ActiveX

כדי ליצור ActiveX DLL, עליך להוסיף את מחלקתך לפרויקט מסוג ActiveX DLL. הדרך הקלה ביותר לבצע זאת הינה ליצור פרויקט מסוג זה ולבחור ActiveX DLL מתוך תיבת הדו-שיח New Project. כזכור לך, לאחר בחירת פרויקט Standard EXE חדש, אתה מקבל פרויקט עם טופס בשם Form1. בצורה דומה, יצירת פרויקט ActiveX DLL או DLL חדש גורמת ל- Visual Basic ליצור פרויקט עם מודול מחלקתי בשם Class1. אם הינך מתחיל לבנות מחלקה חדשה, עליך רק לשנות את Class1 לשם טוב יותר ולהתחיל להזין קוד. אם ברצונך להוסיף לפרויקט מודולים מחלקתיים אותם כבר כתבת, לחץ לחיצה ימנית בחלון Project Explorer ובחר Add file.

הערה:

פרויקטים מסוג ActiveX DLL או EXE מסוגלים להכיל מספר מודולים מחלקתיים, וכן מודולי קוד וטפסים. אך רק האובייקטים בתוכם המוגדרים כציבוריים זמינים ליישומים אחרים

עבודה עם מספר פרויקטים

בחלק הראשון של פרק זה, הוספת מודול מחלקתי לפרויקט Standard EXE. אומנם המודול עמד ברשות עצמו, אך הוא עדיין היווה, יחד עם הטופס, חלק מאותו פרויקט. עם זאת, כאשר תעבוד עם פרויקט ActiveX DLL, רוב הסיכויים יהיו שתהיה זקוק לפרויקט נפרד מסוג Standard EXE כדי לבדוק את מחלקותיך, מכיון ש- ActiveX DLL מיועדים לשימוש מתוכנית חיצונית. במילים אחרות, ממשק המשתמש של היישום יכול להיות פרויקט אחד של Visual Basic, ומחלקת הגישה לנתונים תהיה פרויקט אחר ונפרד. כדי לפתח ולבדוק ביעילות יישום כמו זה, יהיה עליך לדמות פעולת גומלין זו. למרבה המזל, ביכולתך לבצע זאת בכמה דרכים, כמפורט להלן:

- **שימוש בקבוצת פרויקטים:** הוסף מספר פרויקטים לחלון Project Explorer.
- **שימוש במספר מופעים של Visual Basic:** הפעל שנית את Visual Basic וטען את פרויקט ActiveX DLL שלך במופע השני של Visual Basic.
- **שימוש ב-DLL המהודר:** השתמש בשיטה זו כדי לשתף מפתחים אחרים במחלקות שלך.

בזמן שאתה קורא סעיף זה, חשוב על מצב בו קיימים שני פרויקטים: פרויקט ActiveX DLL המכיל מחלקה, ופרויקט Standard EXE המשתמש במחלקה זו.

הערה:

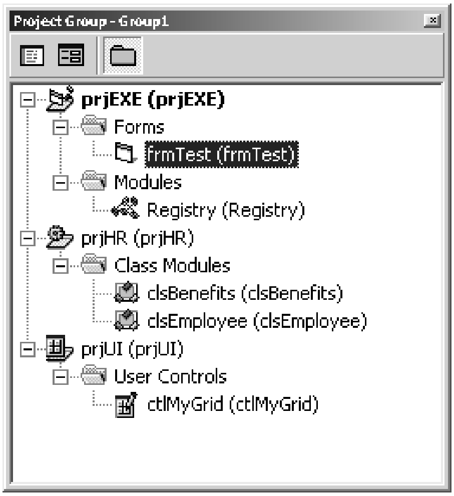
כאשר הינך ניגש ל- ActiveX DLL מתוך פרויקט ActiveX EXE באחת מהדרכים המוסברות לעיל, עליך תמיד להוסיף הפניה אל DLL באמצעות תיבת הדו-שיח References. ראה את הסעיף "שימוש ופנייה אל DLL", מאוחר יותר בפרק זה.

שימוש בקבוצת פרויקטים

באפשרותך ליצור קבוצת פרויקטים ב- Visual Basic על ידי בחירה ב- Add Project מתפריט File. כל פרויקט בקבוצה יופיע בחלון Project Explorer, כפי שניתן לראות בתרשים 16.4

הערה:

יצירת קבוצת פרויקטים הינה שונה מהדרך הרגילה בה מטפלים בפרויקטים מרובים של Visual Basic. באופן רגיל תשתמש באופציית התפריט Open Project, שתגרום לפרויקט הנבחר להחליף את כל הפרויקטים הפתוחים כרגע. בזמן יצירה של קבוצת פרויקטים, מספר פרויקטים נטענים לסביבת הפיתוח **באותו זמן.**



תרשים 16.4: קבוצת פרויקטים מאפשרת לבדוק ולנפות שגיאות ברכיבי ActiveX DLL בתוך מופע יחיד של Visual Basic

הדרך הרגילה לשימוש בקבוצות פרויקטים היא: ראשית, ליצור מרכיב ActiveX, ולאחר מכן להוסיף פרויקט Standard EXE כדי לבדוק את אותו מרכיב. הינך יכול להשתמש בכלים לניפוי שגיאות המסופקים על ידי Visual Basic, כמו Stepping ושימוש בנקודות שבירה, בכל הפרויקטים הנמצאים בקבוצה.

שים לב שלפני הלחיצה על לחצן ההתחלה של Visual Basic, תצטרך לציין את הפרויקט ההתחלתי מתוך קבוצת הפרויקטים. הפרויקט ההתחלתי לפי ברירת המחדל הוא הפרויקט הראשון שנפתח, ושמו מופיע באותיות מודגשות בחלון Project Explorer. כדי לבחור פרויקט התחלתי אחר, לחץ לחיצה ימנית על שם הפרויקט הנבחר, ובחר Set as Start Up.

הערה:
ברוב המקרים יהיה עליך להגדיר את הפרויקט Standard EXE בתור הפרויקט ההתחלתי. גם פקדים וספריות DLL יופעלו, אך רק הפרויקט ההתחלתי יקבל את הפוקוס.

שימוש במספר מופעי Visual Basic

לפעמים, שימוש במספר מופעים של Visual Basic הינו נוח יותר משימוש בקבוצת פרויקטים. אם יש לך פרויקטים מאוד מסובכים של EXE או DLL, שימוש במספר מופעי Visual Basic יכול להוות אפשרות טובה יותר, כיון שבמקרה זה תוכל להביט בכל פרויקט כאשר הוא נמצא בסביבה משלו. ההבדל העיקרי הוא שבקבוצת פרויקטים, Visual Basic מטפלת בהפעלה ובעצירה של כל הפרויקטים. שימוש במספר מופעי Visual Basic דורש ממך להפעיל ידנית את הפרויקטים הנחוצים לפרויקט נתון. במילים אחרות, יהיה עליך להפעיל ראשית את פרויקט הספריה, ולאחר מכן את הפרויקט Standard EXE.

כל תכונות ניפוי השגיאות עדיין תהיינה זמינות במופעים השונים של Visual Basic. במילים אחרות, תוכל להגדיר נקודות שבירה ולצעוד קדימה ואחורה בקוד בשני הפרויקטים. עם זאת, כיון שתקשורת בין-תהליכית נוספת מעורבת בין המופעים השונים של Visual Basic, יהיה עליך להתחיל מחדש פרויקטים לעיתים קרובות יותר מאשר עם קבוצת פרויקטים.

שימוש ב-DLL המהודר

השימוש ב- ActiveX DLL מהודר מאפשר מופע יחיד של Visual Basic. דרך זו אינה מתאימה לבדיקת קוד השייך למחלקה, כיון שאין דרך להשתמש באף אחת מתכונות ניפוי השגיאות. עם זאת, מפתחים אחרים יוכלו להשתמש ב-DLL המהודר על מחשביהם, מבלי שיהיה עליהם להעתיק את הפרויקט והקוד שלך. שימוש ב-DLL מהודר מצריך מעט שלבים פשוטים:

1. **התקנה:** העתק את ה-DLL ואת המרכיבים או הקבצים הנחוצים בשבילו (אם יש כאלה) אל המחשב של המפתח.
2. **רישום ה-DLL:** השתמש בתוכנית REGSVR32 כדי לרשום את הקובץ במערכת.
3. **הוספת הפניה ל-DLL:** לאחר התקנת ה-DLL ורישומו, עליו להופיע בתיבת הדו-שיח References.

פנייה ושימוש ב-DLL: לא משנה באיזו דרך תשתמש כדי לטעון את פרויקט ActiveX DLL (קבוצת פרויקטים, מספר מופעי Visual Basic או DLL מהודר), תמיד יהיה עליך **להפנות** (Reference) אל DLL בפרויקט Standard EXE. הצג את תיבת הדו-שיח References על ידי בחירה באפשרות References מתוך התפריט Project. רשימת ההפניות הזמינות צריכה להכיל את שם פרויקט ActiveX DLL שלך. לאחר שתסמן את התיבה ליד שם ה-DLL, הוא יהפוך לזמין עבור הפרויקט שלך.

הערה:

בזמן שימוש במספר מופעי Visual Basic, יכול להיות שיהיה עליך להוסיף מחדש הפניה אל ה-DLL לאחר שתבצע בו שינויים. בנוסף לכך, הפרויקט ActiveX DLL לא יופיע בתיבת הדו-שיח References לפני תחילת הפרויקט.

גישה למחלקה: בדוגמה הקודמת, יצרת מופע חדש של אובייקט העובד, פשוט על ידי כתיבת שם המחלקה:

```
Dim objBoss As New clsEmployee
```

אך כאשר המחלקה מוגדרת במרכיב ActiveX מהודר וחיצוני, עליך לכלול גם את שם הפרויקט:

```
Dim objBoss As New EmployeeDLL.clsEmployee
```

אם יש לך מספר מחלקות ציבוריות המוגדרות בתוך DLL, תוכל לגשת לכולן:

```
Dim objCEO As New EmployeeDLL.clsExecutive
Dim objDilbert As New EmployeeDLL.clsEngineer
```

שים לב שגם אם תכתוב קוד המשתמש במחלקות חיצוניות אלו, כל תכונות העורך (כמו רשימות פרמטרים) תהיינה זמינות בחלון העריכה של Visual Basic.

קביעת המאפיין הופעה

בפרויקטים של ActiveX DLL או ActiveX EXE, לחלון המאפיינים של המודול המחלקתי מופיע מאפיין נוסף: **הופעה** (Instancing), אשר אינו קיים בפרויקט Standard EXE. מאפיין זה קובע כיצד ייווצרו מופעי המחלקות. האפשרויות משתנות בהתאם לסוג הפרויקט. בזמן למידת יסודות ה-OOP והמחלקות, אתה בהחלט יכול לשכוח ממאפיין זה ולבצע ניסויים בפרויקט Standard EXE. עם זאת, מאפיין זה בדרך כלל נקבע בזמן היווצרות המחלקה, ולכן הוא מאוזכר כאן. טבלה 16.1 מסבירה אודות הגדרות המאפיין Instancing:

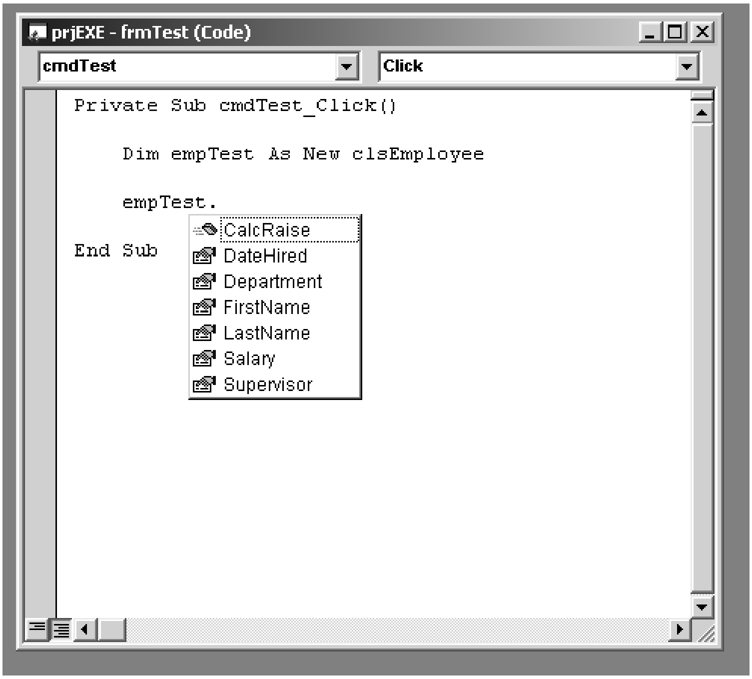
טבלה 16.1: שימוש במאפיין **Instancing**

ערך	שם	רכיבים	הסבר
1	Private	EXE & DLL	לא ניתן לגשת למחלקה זו מחוץ לפרויקט הנוכחי
2	PublicNotCreatable	EXE & DLL	מופעי מחלקה זו יכולים להיווצר רק מתוך הפרויקט אשר מגדיר את המחלקה. עם זאת, יישומים אחרים יכולים לגשת למופעי המחלקה, לאחר היווצרותם
3	SingleUse	EXE בלבד	בכל פעם שנוצר מופע חדש של המחלקה, מופעל עותק חדש של EXE המכיל את המחלקה
4	GlobalSingleUse	EXE בלבד	כמו SingleUse, אך הוא גורם למאפיינים ולשיטות להופיע כמשתנים ושגרות ציבוריים בתוכנית הלקוח (Client)
5	MultiUse	EXE & DLL	יישומים אחרים יכולים ליצור כל מספר של מופעי מחלקה, בזמן שרק עותק אחד מהתוכנית המכילה את המחלקה מופעל
6	GlobalMultiUse	EXE & DLL	כמו MultiUse, אך הוא גורם למאפיינים ולשיטות להופיע כמו משתנים ושגרות ציבוריים בתוכנית הלקוח

המידע בטבלה יכול להיות מובן יותר אם חושבים על ההבדלים בין קבצי EXE וקבצי DLL. קובץ ActiveX DLL מיועד לשימוש בתור רכיב משותף, כך שלקוחות מרובים מסוגלים לגשת לאותו מופע של רכיב זה, ואילו EXE רץ במרחב זיכרון משלו. בנוסף לכך, כאשר מדובר ברכיב מסוג ActiveX EXE, משתמשים יכולים להריץ את התוכנית, ובאותו זמן תוכניות אחרות יכולות לפנות אליו ולהשתמש במחלקות מתוכו.

מספור (Enum)

אחת התכונות המצוינות של Visual Basic היא הדרך שבה היא מציגה מידע ורשימות בהקלדת קוד. לדוגמה, אם הינך משתמש במחלקה clsEmployee, Visual Basic מסוגלת להציג את רשימת המאפיינים והשיטות הזמינים, כמוצג בתרשים 16.5.



תרשים 16.5: תכונות כמו הצגה אוטומטית של חברי המחלקה וסורק האובייקטים, פועלות על מחלקות שנוצרו על ידך

עם זאת, אם כתבת שיטה בתוך המחלקה, לעיתים קרובות יהיה עליך למצוא דרך לדעת את ערכי הפרמטרים התקפים. דוגמה אחת לכך הינה קבועי תיבת ההודעה (כמו vbCritical) אשר מוצגים בזמן שהינך מקליד את הפקודה MsgBox. חשוב על דוגמת המאפיין Department במחלקה clsEmployee. למרות שהינך מאפשר למשתמשים להציב מספר ערכי מחרוזת, בפועל נשמרת מחלקת הפועל בתור מספר שלם. בנוסף לכך, כל עוד אין למשתמשי המחלקה רשימת מחלקות תקפות, הם לא יוכלו לדעת כי MARKETING הוא ערך בר-תוקף.

דרך אחת לפתור את הבעיה הזו היא להשתמש במספור, או לפי Visual Basic, Enum. המספור מאפשר לך להגדיר קבוצת קבועים בתור סוג מידע. באפשרותך להשתמש בסוג מידע זה בתור פרמטרים של פונקציות.

לדוגמה, תוכל להגדיר מספור למחלקות עובדים, ולאחר מכן להגדיר מחדש את המאפיין Department של clsEmployee, כמוצג בקוד הבא.

```
Public Enum Departments
	Marketing = 101
	Accounting = 102
	InformationTech = 103
	Unknown = 104
End Enum

Dim nDeptNum As Integer

Public Property Get Department() As Departments
	Department = nDeptNum
End Property

Public Property Let Department(ByVal NewDept As Departments)
	nDeptsNum = NewDept
End Property
```

לאחר מכן תוכל לגשת למאפיין Department כדלקמן:

```
objEmployee.Department = InformationTech
If objEmployee.Department = InformationTech Then MsgBox "IT department"
```

תוכל לבדוק את זה על ידי הוספת סעיף Enum לפרויקט לדוגמה אשר יצרת מוקדם יותר. שים לב לכך שכאשר תכתוב את סמל השוויון כדי להציב או לקרוא את ערך המאפיין Department, הערכים תקפים יוצגו לפניך בתוך רשימה נפתחת.

יצירת מחלקות המכילות אוספים

אוסף (Collection) הוא קבוצת פריטים, אשר הם בעצמם סוג אובייקט. ב- Visual Basic קיימים מספר אובייקטי-אוספים, כמו Forms ו-Controls. ניתן להשתמש בפקודה For Each...Next עם אוספים כדי לבצע פעולות על כל האובייקטים הכלולים באוסף. השיגרה MinimizeAll, המוצגת בתוכנית 16.1, ממזערת את כל הטפסים הטעונים ביישום.

תוכנית 16.1: MINIMIZE.TXT - שימוש במבנה For Each לטיפול בכל פריטי האוסף

```
' Minimize all loaded forms.
Sub MinimizeAll()
	Dim frmElement As Form
	' For each loaded form.
	For Each frmElement In Forms
		'Minimize the form.
		frmElement.WindowState = vbMinimized
	Next frmElement
End Sub
```

האוספים פותרים 3 בעיות עיקריות אשר בהן נתקלים רוב המפתחים בזמן העבודה עם אובייקטים:

- הם מספקים דרך סטנדרטית לגישה ומעקב אחר מספר מופעי אובייקטים.
- הם מקבצים מספר אובייקטים מקורבים למטרות קבועות, כמו שינוי מאפייני הצבעים, או גרירה למיקום חדש.
- בניגוד למערך, האוסף יכול להכיל חברים מסוגים שונים.
- הם מארגנים מערכות גדולות של אובייקטים לצורה של היררכיה.

הסעיפים הבאים מסבירים כל אחד מהיבטי השימוש באוספים, אשר בהם תיתקל כאשר תיצור יישומים מבוססי אובייקטים ב- Visual Basic.

שיטות ומאפיינים סטנדרטיים באוספים

אוספים המוגדרים על ידי המשתמש חולקים קבוצה זהה של מאפיינים ושיטות. האוספים יכולים להכיל מאפיינים ושיטות נוספים, אך כל האוספים שתיצור ב- Visual Basic יכילו לפחות את הקבוצה המופיעה בטבלה 16.2.

הערה:

אוספים המוגדרים מראש על ידי Visual Basic עלולים שלא להכיל את כל המאפיינים והשיטות המופיעים להלן:

טבלה 16.2: מאפיינים ושיטות המשותפים לכל האוספים

פריט	שימושו
המאפיין Count	מדווח את מספר האובייקטים באוסף
המאפיין Item	מספק גישה לאובייקט יחיד מהאוסף

בנוסף לפריטים בטבלה 16.2, רוב האוספים בדרך-כלל מספקים עוד שתי שיטות מקובלות, המופיעות בטבלה 16.3

טבלה 16.3: שיטות המקובלות ברוב האוספים

שיטה	שימושה
Add	מוסיפה אובייקט לאוסף
Remove, Delete	מוחקת אובייקט מהאוסף

השיטות Add ו-Remove מספקות למפתחים דרך סטנדרטית ליצור ולמחוק אובייקטים מהאוסף. השיטה Add כוללת לרוב מפתח ואינדקס המועברים כפרמטרים, ומשמשים לזהות את האובייקט בתוך האוסף. עם זאת, אוספים שונים עשויים ליישם את

השיטות Add ו-Remove בדרכים שונות. לדוגמה, השווה בין תחביר Add של האוסף ListItems השייך לפקד ListView, ובין האוסף ListItems של הפקד ImageList :

```
ListView1.ListItems.Add index,key,text,IconNumber,smallIconNumber
ImageList1.ListImages.Add index,key,picture
```

יצירת אוסף חדש לפעולות קבוצתיות

הינך יכול ליצור אוספים כדי להכיל טפסים, פקדים ואובייקטים. השתמש באובייקט Collection כדי ליצור אוסף חדש. ההצהרה הבאה יוצרת אוסף חדש בשם colSelected :

```
Dim colSelected As New Collection
```

הצהרת משתנה בתור אובייקט מסוג Collection נותנת לאובייקט שלך 4 מאפיינים ושיטות מובנים, כפי שניתן לראות בטבלה 16.4.

טבלה 16.4: מאפיינים ושיטות מובנים באובייקט Collection

פריט	תפקיד
המאפיין Count	מחזיר את מספר האובייקטים אשר באובייקט
השיטה Add	מוסיפה אובייקט לאוסף
המאפיין Item	מספקת גישה לאובייקט יחיד מהאוסף
השיטה Remove	מוחקת אובייקט מהאוסף

הקוד בתוכנית 16.2 יוצר אוסף חדש בשם colTextBoxes ומוסיף את כל תיבות הטקסט שעל הטופס לאוסף החדש.

תוכנית 16.2: COLLECT.TXT - יצירת אוסף בעזרת קוד זה

```
Option Explicit

' Create a new collection to contain all the
' text boxes on a form
Dim colTextBoxes As New Collection

Private Sub Form_Initialize()
	' Variable used in For Each to get controls.
	Dim cntrlItem As Control
	' Loop through the controls on the form.
	For Each cntrlItem In Me.Controls
```

```
            ' If the control is a text box, add it to the
            ' collection of text boxes.
            If TypeOf(cntrlItem) = "TextBox" then
                colTextBoxes.Add cntrlItem
            End If
        Next cntrlItem
End Sub
```

הקוד בתוכנית 16.3 משתמש באוסף colTextBoxes כדי לנקות את כל הטקסט אשר הוקלד על הטופס.

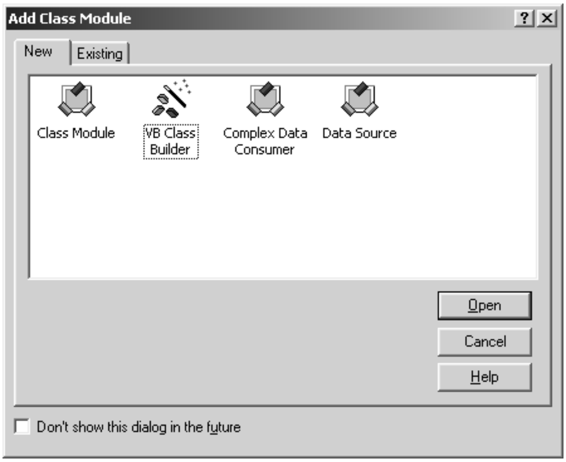
תוכנית 16.3: CLEAR.TXT - שימוש בלולאת For...Each כדי לטפל באוסף

```
Sub cmbClear_Click()
        ' Variable used in For Each to get controls
        Dim cntrlItem As Control
        ' Clear each of the text boxers in the collection.
        For Each cntrlItem In colTextBoxes
            cntrlItem.Text = ""
        Next cntrlItem
End Sub
```

ראה: "שימוש בפקד ListView", **פרק 12**.

שימוש באשף המחלקות

כאשר הינך יוצר מחלקה חדשה, Visual Basic מציגה לך את האפשרויות (בין היתר): ליצור מחלקה ריקה, או להפעיל את **אשף המחלקה** (Class Builder).

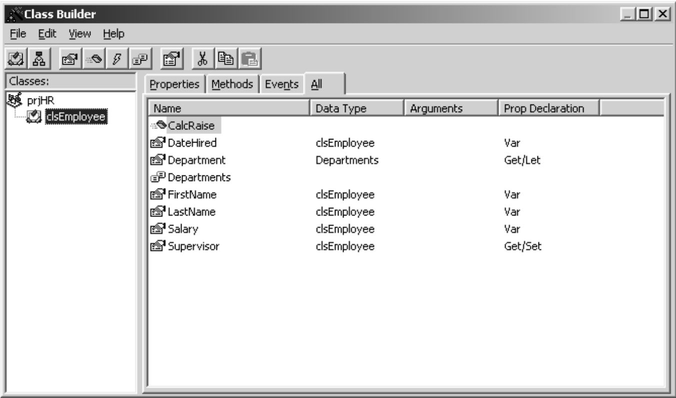


תרשים 16.6: לחיצה כפולה על אשף המחלקה מפעילה כלי העוזר לארגן מחלקות

הערה:

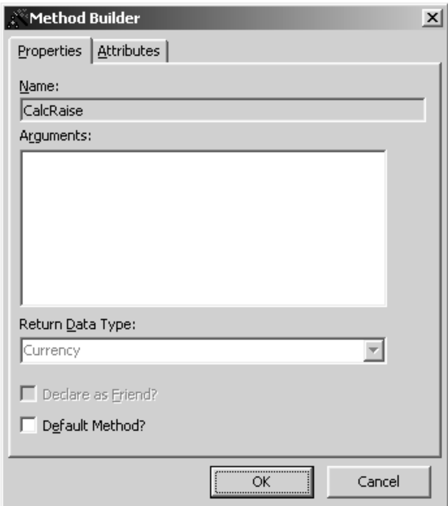
אם אינך רואה את Class Builder כמוצג בתרשים 16.6, השתמש במנהל התוספות (Add-in Manager, הנמצא בתפריט Add-ins) כדי להפוך את האשף לזמין.

ניתן להשתמש באשף המחלקה כדי ליצור קווי מתאר כלליים עבור המחלקות החדשות, או לארגן מחדש מחלקות קיימות. תרשים 16.7 מציג את אשף המחלקה עם המחלקה clsEmployee טעונה בתוכו.



תרשים 16.7: אשף המחלקה מספק ממשק גרפי למחלקות, מאפיינים, שיטות, קבועים ואירועים

כדי להוסיף פריט מסוים לפרויקט שלך, לחץ על אחד מלחצני סרגל הכלים. אשף המחלקה יציג תיבת דו-שיח ובה תוכל לכוון את תכונותיו של הפריט החדש. תרשים 16.8 מציג את תיבת הדו-שיח המוצגת כאשר מוסיפים שיטה חדשה.



תרשים 16.8: תיבת הדו-שיח של אשף השיטה מציגה את כל האפשרויות הזמינות

כאשר תסגור את אשף המחלקה, הוא יעדכן את הפרויקט שלך בכל השינויים או ההוספות שביצעת. עדיין יהיה עליך להוסיף קוד כדי שמחלקתך תבצע פעולות כלשהן, אך אשף הביטויים ייצור את ההצהרות הדרושות ואת הקוד התומך.

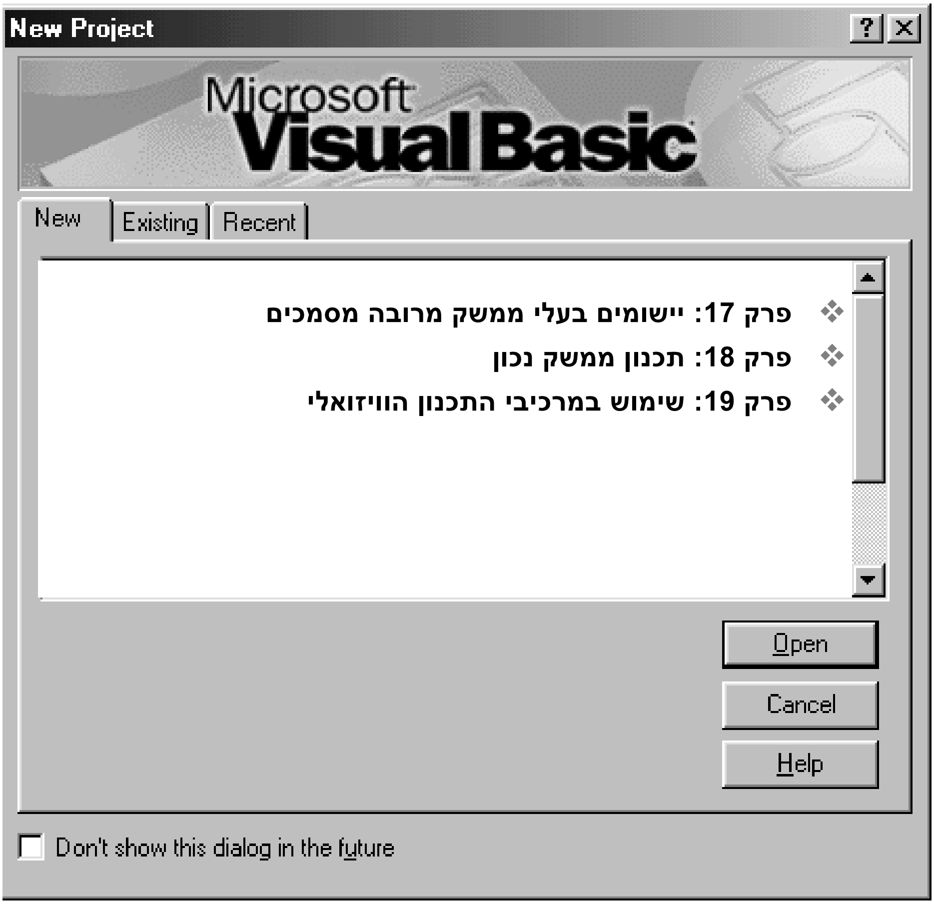
מכאן...

בפרק זה למדת לבנות מודולים מחלקתיים, והיכרת את האובייקטים אשר ניתן ליצור מהם. כדי ללמוד עוד על כמה מהנושאים המכוסים בפרק זה, פנה לפרקים הבאים:

- כדי לראות כיצד אוספים מתפקדים יחד עם פקדים, ראה פרק 12 **הפקדים המשותפים של Microsoft**.
- הינך יכול ללמוד כיצד ניתן להשתמש בטכניקות המוחלות על מחלקות כדי לבנות פקדי ActiveX משלך באמצעות עיון בפרק 14, **יצירת פקדי ActiveX**.
- כדי ללמוד עוד אודות גישה לאובייקטים ואוספים מיישומים אחרים, כמו Word או Excel, ראה פרק 22 **שימוש ב-OLE לשליטה על יישומים אחרים**.
- אוספים יכולים לעזור גם בתחום ה-Web, כמתואר בנספח 4 **דפי שרת פעילים**.

חלק 4

ממשקי Visual Basic



17

יישומים בעלי ממשק מרובה מסמכים

מה בפרק זה?

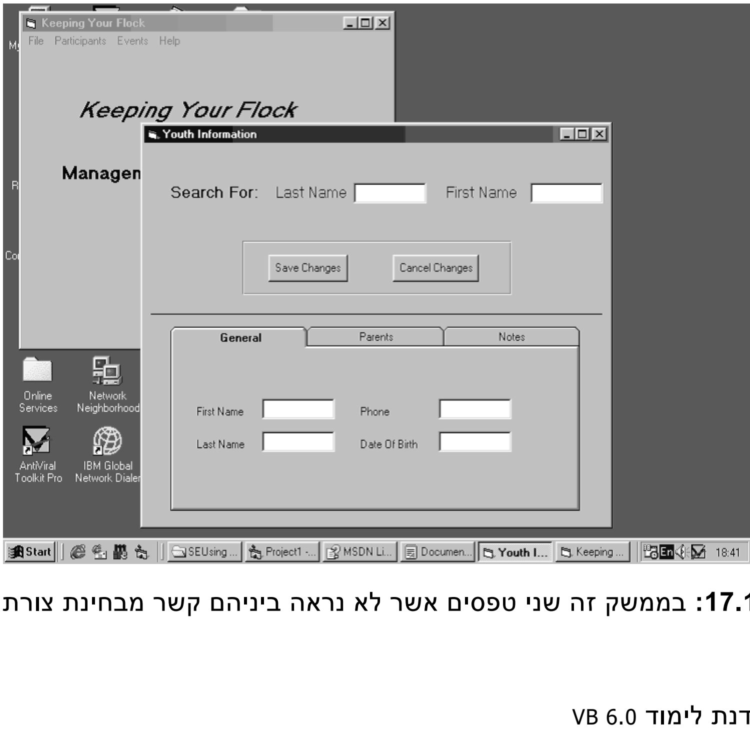
- מבוא ליישומי MDI
- יצירת תוכנית MDI פשוטה
- יצירת עותקים מטופס בודד
- עבודה עם תפריטים
- טיפול בטפסים צאצאים
- יצירת תוכנית דוגמה לניהול פגישות הפועלת בסביבת MDI

כאשר תתחיל לכתוב תוכניות מורכבות יותר ב- Visual Basic, בוודאי תרצה להשתמש בממשק ריבוי המסמכים של Windows. ממשק זה מאפשר לתוכניות לעבוד עם מספר מסמכים הכלולים ב**טופס אב** (Parent Form) בודד. השימוש בטכניקת MDI מאפשר יצירת ממשק ברור ומובן יותר בהשוואה לממשקים הכוללים טפסים עצמאיים ובלתי תלויים המפוזרים על גבי המסך.

תקן MDI יכול לשפר את תוכניתך בשתי דרכים אפשריות. ראשית, תוכל להשתמש בטופס אב אחד כרקע לכלל היישום שלך. בממשק קיים טופס האב (Container Form) ואילו הטפסים הנוספים, המונחים על גביו, מכונים **טפסי צאצא** (Child Forms). כאשר המשתמש גורר או מזיז את טופס האב, טפסי הצאצא זזים גם הם, דבר המסייע לשמור על ממשק התוכנית שלך נקי ומסודר. שנית, ואף חשוב מכך, תוכנית MDI מאפשרת למשתמשיה לעבוד עם מספר טפסים בו-זמנית. כמו כן, תוכניות MDI מאפשרות עבודה בו-זמנית עם מספר מופעים של אותו טופס, דבר היכול להוסיף גמישות ועוצמה רבה ליישום שלך.

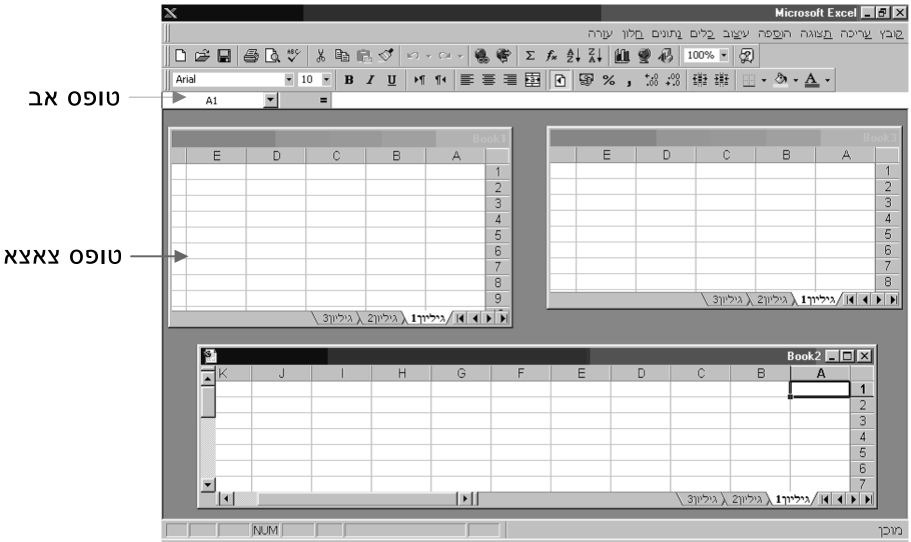
מבוא ליישומי MDI

רבים מהיישומים שאתה יוצר ב- Visual Basic כוללים סדרת טפסים עצמאיים, דומים לאלה המוצגים בתרשים 17.1. כל אחד מטפסים אלה מופיע על המסך בנפרד וניתן להזיזו, להגדילו או להקטינו ללא תלות בטפסים אחרים המצויים אותה עת על המסך. בסוג כזה של ממשק אומנם קשה לארגן את הטפסים בקלות או לטפל בהם כקבוצה, אך אפילו אם נתחשב במגבלה זו, ממשק זה הוא עדיין הנפוץ ביותר מבין הממשקים הקיימים ומתאים לעבודה עם סוגים רבים של תוכניות.



תרשים 17.1: בממשק זה שני טפסים אשר לא נראה ביניהם קשר מבחינת צורת הצגתם

דרך אלטרנטיבית לממשק הסטנדרטי שהוצג בתרשים 17.1, היא שימוש ב**ממשק מרובי מסמכים**, או **MDI** (Multiple Document Interface). בסוג זה של תוכניות קיים טופס אב אחד אשר מכיל את מרבית הטפסים האחרים בתוכנית. הטפסים האחרים יכולים להיות טפסי צאצא, הנמצאים בתוך טופס האב, או טפסים רגילים שאינם נמצאים בו. בעזרת יישום מסוג MDI, תוכל לגרום בקלות את כל טפסי הצאצא או למזער את כל הטפסים על ידי מזעור יחיד של טופס האב. יישומים כגון Microsoft Word ו-Excel הם דוגמאות ליישומי MDI. אם עבדת עם תוכניות אלו בעבר, אתה בוודאי יודע שניתן לפתוח מספר חלונות בתוכנית בעת ובעונה אחת, לגשת אליהם בקלות דרך תפריט חלון ולמזער את כולם בעזרת לחיצה אחת של העכבר. גם Visual Basic עצמה עברה שינוי מראה והפכה ליישום MDI החל מגירסה 5.0. תרשים 17.2 מציג שלושה גליונות עבודה הנפתחים בעת ובעונה אחת ב-Excel, כדוגמה ליישום MDI אופייני.



תרשים 17.2: יישומי MDI מאפשרים לך לטפל בקלות בריבוי חלונות מסמך

תכונות טפסי אב

טופס MDI, הידוע גם בכינויו **טופס אב** (Parent Form), משמש כשטח אחסון לכל הטפסים הצאצאים בתוכנית. לטופס MDI יש מספר תכונות המגדירות את התנהגותו:

- בכל תוכנית יכול להיות טופס MDI אחד בלבד.
- טופס MDI יכול להכיל רק את הפקדים אשר תומכים במאפיין Align, כגון פקד PictureBox או ToolBar. אינך יכול להציב פקדים אחרים בטופס MDI.
- אין אפשרות להשתמש בפקודת Print או בכל אחת מהפקודות הגרפיות כדי להציג מידע בטופס MDI.

- חלון האב וכל חלונות הצאצאים מיוצגים על ידי סמל (Icon) בודד בשורת המשימות של Windows. אם טופס האב ממוזער ומוחזר לגודלו הקודם לאחר מכן, כל הטפסים הצאצאים יוחזרו לאותה צורת התצוגה שהיתה להם לפני שהתוכנית מוזערה.
- אם פריט בתפריט מוגדר עבור טופס צאצא, שם התפריט יוצג בשורת התפריטים של טופס האב. אם מוגדר תפריט לטופס האב, הוא לא יוצג אם לטופס הצאצא הפעיל יש תפריט משלו.

תכונות טפסי הצאצא

כמו שלטופס MDI יש תכונות התנהגות משלו, גם לטפסי צאצא יש צורת התנהגות מסוימת. תכונות טפסי הצאצא הן:

- כל טופס צאצא יוצג בתוך גבולות טופס האב. אין אפשרות להוציא את טופס הצאצא אל מעבר לגבולות טופס האב.
- כאשר טופס צאצא ממוזער, הסמל שלו יוצג בחלון האב ולא בשורת המשימות של Windows.
- כאשר טופס צאצא מוגדל לגודלו המקסימלי, הוא ממלא את כל השטח הפנימי של טופס האב. בנוסף, כותרת טופס האב מכילה את שם טופס האב ואת שם טופס הצאצא המוגדל.
- כאשר טופס צאצא מוגדל לגודלו המקסימלי, כל הטפסים הצאצאים האחרים מוגדלים גם הם.

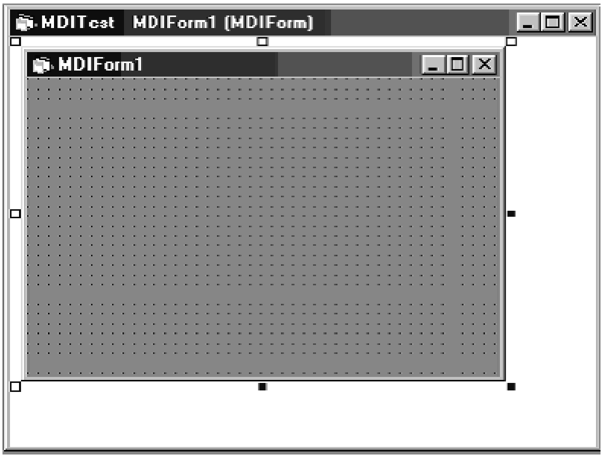
יצירת תוכנית פשוטה בעלת ממשק MDI

כמו בשיטות תכנות רבות, הדרך הטובה ביותר להבין כיצד פועלים יישומי MDI היא ליצור תוכנית פשוטה. סעיף זה יעביר אותך דרך תהליך הקמת "מעטפת" לתוכנית MDI. התוכנית תורכב מטופס MDI וטופס צאצא יחיד. לאחר מכן תוכל להשתמש בתוכנית כבסיס ליישומי MDI מלאים.

כשלב ראשון צור פרויקט חדש ב- Visual Basic על ידי בחירת אפשרות **New project** מתוך תפריט **File**.

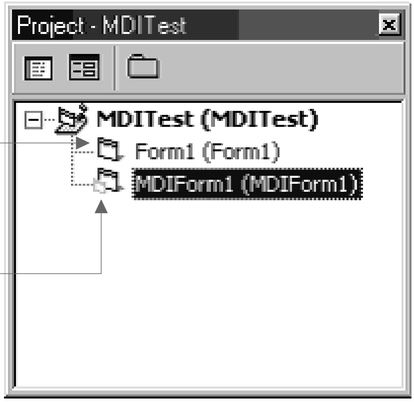
יצירת טופס אב

הצעד הבא הוא ליצור את טופס האב (MDI). כדי ליצור טופס MDI לפרויקט שלך, בחר בתפריט **Project**, הוסף טופס MDI, או בחר בטופס MDI מתוך התפריט הנפתח שמופיע לאחר לחיצה על הלחצן **MDI Object**. לאחר מכן, בחר את סמל טופס MDI מתוך תיבת הדו-שיח **Add MDI Form** ובחר **Open**. כאשר טופס MDI יתווסף לפרויקט, הוא ייראה כמו הטופס שבתרשים 17.3.



תרשים 17.3: טופס MDI הינו בעל רקע כהה יותר מאשר טופס רגיל

טופס MDI יתווסף לתיקיית הטפסים בחלון סייר הפרויקט (Project Explorer Window). אולם, אם תסתכל במבט בוחן, תוכל לראות כי הסמל המשויך לטופס MDI בחלון הפרויקט, שונה מהסמל המשויך לטופס רגיל.



תרשים 17.4: ניתן להבחין בסוג הטופס בחלון הפרויקט לפי הסמל שלו

לאחר הוספת הטופס לפרויקט, עליך לספק שם עבור הטופס ולקבוע את המאפיינים הדרושים. מרבית המאפיינים של טפסי MDI זהים לאלה של טפסים רגילים.

ראה: "חלקי הטופס", פרק 3.

קיימים שני מאפיינים אשר הינם ייחודיים לטפסי MDI וראויים להתייחסות מיוחדת: מאפיין AutoShowChildren ומאפיין ScrollBars. מאפיין AutoShowChildren קובע אם טפסי צאצא יוצגו אוטומטית בזמן טעינתם. אם ערך המאפיין הוא True (ערך ברירת המחדל), הטפסים הצאצאים יוצגו מייד לאחר שייטענו לזיכרון. מכאן ניתן להבין כי פקודת Load ושיטת Show הן בעלות השפעה זהה על הטופס.

מאפיין ScrollBars קובע האם, במידת הצורך, יופיעו פסי גלילה בטופס MDI האמור. כאשר ערך המאפיין הוא True (ערך ברירת המחדל), יוצגו פסי גלילה על גבי טופס אם אחד או יותר מטפסי הצאצאים חורג מתחומי טופס האב, כפי שמתואר בתרשים 17.5. אם המאפיין נקבע כ-False, לא יוצגו פסי גלילה.



תרשים 17.5: פסי גלילה מאפשרים להציג חלקי טופס צאצא החורגים מגודל טופס האב

מאפיין חשוב נוסף הוא Picture. למרות העובדה שאין תמיכה בטפסי MDI לשיטת Print ולשיטות גרפיות כמו בטופס רגיל, תוכל עדיין לכלול תמונה כרקע לטופס.

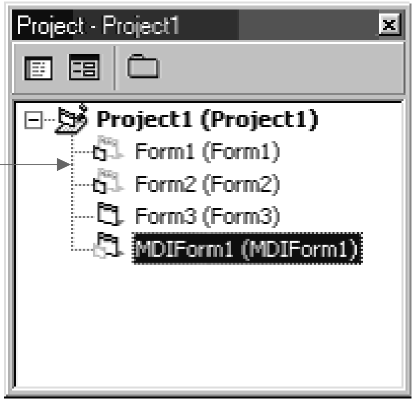
יצירת טופס צאצא

תהליך יצירת טופס צאצא בתוכנית MDI אף פשוט יותר מתהליך יצירת טופס האב. טופס צאצא הוא למעשה טופס רגיל אשר ערך מאפיין MDIChild שלו הוא True. לפיכך, כל מה שכבר ידוע לך על יצירת טפסים רגילים נכון גם לגבי יצירת טפסי צאצא ביישומי MDI.

עבור תוכנית הדוגמה, כל שעליך לעשות הוא לשנות את ערך המאפיין MDIChild של הטופס אשר נוצר. לשם כך, בחר את הטופס בחלון הפרויקט, סמן את מאפיין MDIChild בחלון המאפיינים ושנה את הערך שלו לערך True. תוכל להבחין כי הסמל בחלון הפרויקט השתנה מטופס רגיל לטופס MDI. זהו השינוי היחיד בו ניתן להבחין מסביבת העיצוב (ראה תרשים 17.6).

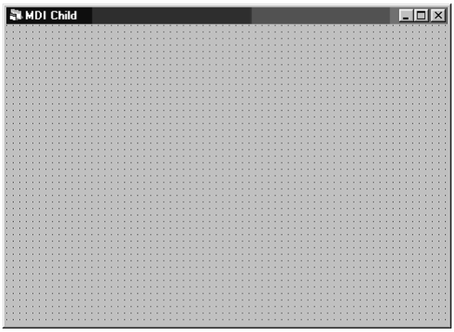
טיפ:

בדומה למאפיינים אחרים בעלי ערכים קבועים מראש, ניתן ללחוץ לחיצה כפולה על מאפיין MDIChild בחלון המאפיינים ולשנות את ערכו בין הערכים הקבועים מראש.

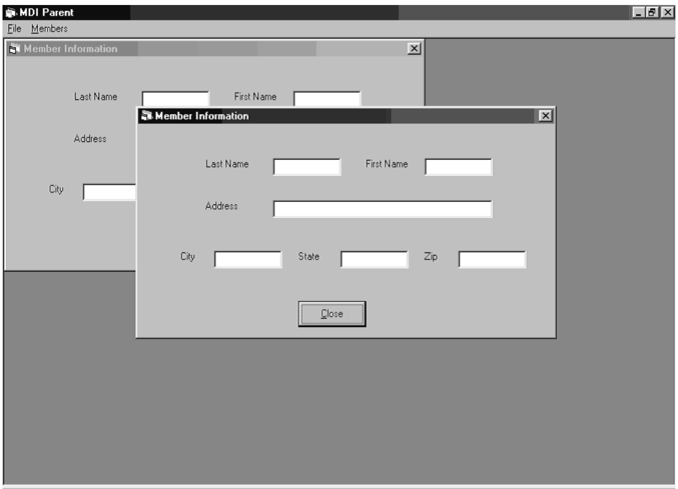


תרשים 17.6: שים לב כי **סמלים** של טפסי הצאצאים שונות בצורתן מ**סמלים** של טפסים רגילים וטופס האב

לאחר שקבעת את ערך מאפיין MDIChild, כל שנשאר הוא להוסיף את הפקדים הדרושים לתוכנית. תוכל, כמובן לעצב את הטופס ואחר לשנות את מאפיין MDIChild. אין חשיבות לסדר הפעולות. בתרשים 17.7 ניתן לראות טופס צאצא טיפוסי.



תרשים 17.7: טפסי צאצא נראים בדיוק כמו טפסים רגילים



תרשים 17.8: יישום MDI פשוט זה מציג טופס אב ושני טפסי צאצא. שים לב שטפסי הצאצאים הם עותקים זהים של אותו טופס

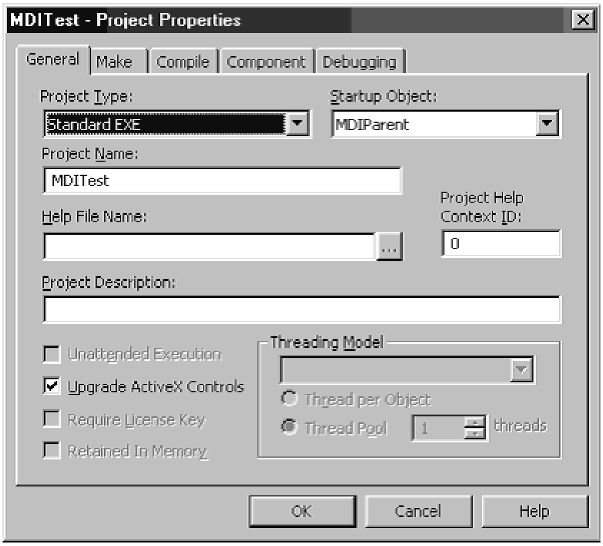
הפעלת התוכנית

לאחר שסיימת לקבוע את טפסי האב והצאצא, תוכל להפעיל את התוכנית כדי לראות כיצד מתנהג טופס צאצא בתוך טופס אב. ראשית, כמו תמיד, עליך לשמור את עבודתך. אחר, לחץ על לחצן Start או הקש F5 כדי להפעיל את התוכנית. כשהתוכנית פועלת, מבנה הטופס אמור להיות דומה לזה המוצג בתרשים 17.8.

כעת בצע את המשימות הבאות כדי להבין את צורת ההתנהגות של טפסי האב והצאצאים:

1. מזער את טופס הצאצא ושים לב למיקום הסמל שלו.
2. גרור את טופס הצאצא על פני המסך. הטופס לא יעבור את גבולות טופס האב.
3. הגדל את טופס הצאצא לגודלו המירבי.
4. מזער והגדל את טופס האב.

בעת הפעלת התוכנית, ודאי שמת לב שטופס הצאצא הופיע מעצמו. הדבר נגרם, בדוגמה הפשוטה שיצרת, משום שטופס הצאצא (אשר נוצר ראשון עם יצירת הפרויקט) עוצב כברירת מחדל, בדומה לטופס הראשי של הפרויקט. אם אתה מעוניין לשנות את מצב ברירת המחדל ולהתחיל את הפרויקט עם טופס אב ריק, עליך לשנות את הגדרות Startup Objects בתיבת הדו-שיח Project Properties, כמתואר בתרשים 17.9. ניתן להגיע למאפייני הפרויקט על ידי בחירת האפשרות **Project_Properties** מתפריט **Project**.



תרשים 17.9: בתיבת דו-שיח זו הינך מגדיר את הטופס ההתחלתי של הפרויקט, כמו גם מאפיינים נוספים

ריבוי מופעים של טופס

ניתן להשתמש בטופס MDI כדי לגרום לתוכניות להיראות טוב יותר ולהקל את הטיפול בטפסים השונים. אולם, אין זו הסיבה העיקרית בגללה כדאי להשתמש ביישומים מסוג זה. המאפיין החזק ביותר והכוח העיקרי של יישומי MDI היא היכולת ליצור ולטפל בריבוי מופעי הטופס בעת ובעונה אחת. לדוגמה, בעבודה עם מעבד התמלילים Microsoft Word, כל מסמך פתוח או חדש מהווה למעשה מופע נוסף מטופס בסיסי קבוע. למעשה, יישומי MDI רבים מכילים שני טפסים בלבד: טופס אב וטופס נוסף המשמש כתבנית (Template) עבור כל טפסי הצאצא ביישום.

הערה:

תוכל ליצור יותר מתבנית אחת עבור הטפסים הצאצאים ביישום שלך. לדוגמה, ל- Visual Basic עצמה יש שתי תבניות בסיסיות של טפסי צאצא: טופס העיצוב וטופס הקוד. תוכל ליצור מספר בלתי מוגבל של תבניות, כל זאת במגבלות המערכת.

יצירת יישום MDI מסוג זה, מחייבת השקעה רבה יותר מזו שנדרשה במסגרת תוכנית הדוגמה. בתחילה עליך להגדיר טופס MDI בסיסי בסביבת העיצוב ולאחר מכן להשתמש במשתני אובייקטים כדי ליצור מופעים של הטופס בזמן פעולת התוכנית.

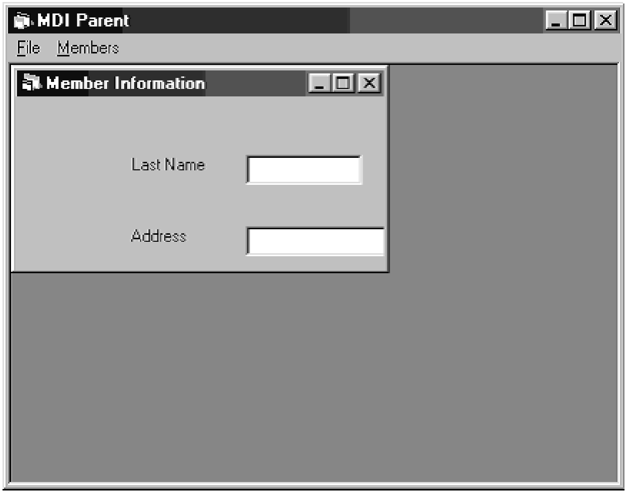
כדי ליצור יישום MDI המכיל מספר מופעים של טופס בודד, התחל פרויקט חדש והוסף לו טופס MDI, כמתואר בסעיף **יצירת תוכנית MDI פשוטה**. לאחר מכן, קבע את טופס MDI כאובייקט ההתחלתי באמצעות תיבת הדו-שיח Project Properties.

הגדרת הטופס הבסיסי

בדומה לסעיף הקודם **ריבוי מופעים של הטופס**, תהליך יצירת **תבנית** (Template) הטופס זהה לתהליך יצירת טופס רגיל. הוסף לטופס את כל הפקדים הדרושים עבור הממשק וכתוב קוד להפעלת הפקדים. בנוסף, עליך לקבוע את ערך המאפיין MDIChild של הטופס לערך True.

בעת הפעלת יישום MDI, תבחין כי כאשר טופס צאצא מופיע בפעם הראשונה, גודלו ומיקומו שונים בהשוואה לאלה הזכורים לך מסביבת העיצוב (ראה תרשים 17.10). הסיבה לכך היא שמערכת ההפעלה Windows מקצה, כברירת מחדל, גודל ומיקום מסוימים לכל טופס צאצא המוצג על המסך.

אם הגודל ומיקום ברירת המחדל אינם מקובלים עליך, תוכל לקבוע את מאפיין הגבול (Border) של החלון לגודל קבוע (Fixed Size) או לחילופין, לכתוב קוד בשגרת Load של טופס הצאצא כדי להציב את הטופס במיקום ובגודל הרצויים לך. תוכל לבדוק את הערכים המוגדרים במאפיינים Height ו-Width של הטופס בסביבת העיצוב ולשלבם בקוד אותו תכתוב בשגרת Load של הטופס. שיטה דומה תוכל לנקוט גם בעת קביעת המיקום הרצוי עבור הטופס: בדוק את הערכים המוגדרים במאפיינים Top ו-Left בסביבת העיצוב, ושלב אותם בשגרת Load. הקוד בתוכנית 17.1 מראה כיצד לקבוע את גודלו של טופס הצאצא ולמרכז אותו בתוך טופס האב.



תרשים 17.10: יישומי MDI קובעים את גודל ומיקום טפסי הצאצא שלהם בצורה אוטומטית

הערה:

הגדלת חלון הצאצא עד כדי חריגה מגבולות חלון האב, כתוצאה מפעולות המשתמש או מפקודות קוד, אומנם אינה מגדילה אוטומטית גם את חלון האב אך גורמת להופעת פסי גלילה המאפשרים צפייה בחלון הצאצא החורג (זאת, כמובן, בהנחה שערך מאפיין ScrollBars הוא True, כפי שכבר למדת בסעיף הקודם, **יצירת טופס אב**).

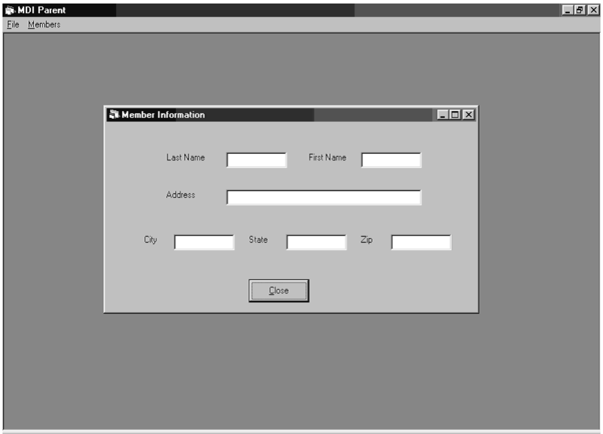
תוכנית 17.1: MDICHILD.TXT שימוש בקוד כדי לקבוע את גודל ומיקום טופס הצאצא.

```
Private Sub Form_Load()
    Me.Height = 2745
    Me.Width = 3690
    Me.Top = (mdiMain.ScaleHeight - Me.Height) / 2
    Me.Left = (mdiMain.ScaleWidth - Me.Width) / 2
End Sub
```

תרשים 17.11 מציג את תוצאות הפעלת הקוד.

הערה:

אינך יכול להשתמש במאפיין StartUpPosition כדי לקבוע את המיקום ההתחלתי של טופס צאצא בחלון MDI. למעשה, אינך יכול לשנות את הגדרות המאפיין עבור טופס צאצא מערך ברירת המחדל שלו Manual - 0.



תרשים 17.11: תוכל לכתוב קוד אשר יבטל את גודל ומיקום ברירת המחדל של טופס צאצא

יצירת טפסים תוך שימוש במשתני אובייקטים

לאחר שיצרת את טופס הצאצא הבסיסי, הינך זקוק לדרך אשר בעזרתה תיצור מופע של הטופס בזמן פעולת התוכנית, ותציג אותו ביישום. לשם כך יש צורך בכתיבת שתי שורות קוד. כהתחלה, תשתמש בפקודת Dim כדי ליצור **משתנה אובייקט** (Object Variable) אשר יכיל מופע של הטופס. במסגרת הפקודה Dim, השתמש במילת המפתח New כדי להורות לתוכנית Visual Basic ליצור **מופע חדש** (New Instance) של הטופס, שכן אחרת תיצור הפקודה כינוי חדש לטופס הקיים. לאחר שיצרת את משתנה האובייקט, עליך להשתמש בשיטת Show כדי להציג את הטופס. במקרה זה, במקום להשתמש בשם הטופס, הגדר את שם המשתנה. שתי שורות הקוד הדרושות לכך הן:

```
Dim NewForm As New MDIChild1
NewForm.Show
```

כדי לראות כיצד הקוד עובד, הצב את שורות הקוד בשגרת Click של טופס MDI והפעל את התוכנית. בכל פעם שתלחץ על טופס MDI, יוצג מופע חדש של טופס הצאצא.

מילות המפתח Me ו-ActiveForm

זה מכבר למדת שכל הטפסים הצאצאים זהים בצורתם ונוצרים על ידי שימוש באותן מילות מפתח, אם כך, כיצד תוכל לבודד טופס מסוים בקוד? בעיה זו מתחדדת כאשר מדובר בקוד כללי היכול לפעול על כל אחד מהטפסים.

כדי להתמודד עם הבעיה, בעת יצירת יישומי MDI, תוכל להיעזר בצמד מילות המפתח: Me ו-ActiveForm. מילות מפתח אלו מאפשרות לך ליצור קוד כללי אשר יוכל לעבוד עם כל טופס צאצא שתיצור.

Me היא מילת מפתח בה ניתן להשתמש בכל טופס לשם התייחסות לטופס עצמו, בדיוק כפי שהמילה **ME** משמשת בשפה האנגלית לזיהוי הדובר מבלי לנקוב בשמו.

בתוכנית 17.1 נתקלת בשימוש במילת המפתח ME לצורך שינוי גודל ומיקום טופס הצאצא בעת טעינתו. שימוש נכון במילת המפתח ME ביחס לשם הטופס לאורך כל הקוד המתייחס לטופס הצאצא, יגרום לקוד שיופעל על כל מופע פעיל בכל זמן נתון.

מילת המפתח ActiveForm היא למעשה מאפיין של טופס האב. השימוש בה דומה לשימוש במילת המפתח Me. ActiveForm מתייחסת לכל טופס צאצא הפעיל בזמן נתון. שימוש נכון במילת המפתח ActiveForm לאורך כל הקוד המתייחס לטופס האב, יגרום לכך שהקוד יופעל על המופע הפעיל בלבד ולא על מופעים אחרים. שורת הקוד הבאה מציגה דוגמה פשוטה לשימוש במאפיין ActiveForm :

```
MDIParent.ActiveForm.Print "This form is currently active"
```

כדי לבדוק את הקוד, הצב אותו בשגרת Click של טופס האב. ודא שקיים יותר מטופס צאצא אחד בזיכרון. לחיצה על אזור ריק בטופס האב תגרום להדפסת הטקסט בטופס הפעיל בזמן הלחיצה.

הערה:

שים לב, כאן ובתוכנית 17.1, לשימוש בקידומת MDI לשם טופס האב.

על ידי שימוש במאפיין ActiveForm, תוכל להתייחס לכל מאפיין, שיטה או אירוע המתרחשים בטופס הצאצא הפעיל בלא צורך בידיעת שם הטופס.

אתחול מופע של טופס צאצא

אם יש צורך בכתיבת קוד אשר יתבצע כאשר מופע של טופס צאצא נוצר, תוכל להציב את הקוד בשגרת Initialize של טופס הצאצא. שיגרה זו מתרחשת בכל פעם שנוצר מופע חדש מהטופס ולפני התרחשות אירוע Load.

לדוגמה, תוכל לקבוע את מאפיין Caption של טופס הצאצא בזמן שהוא נוצר. פרויקט הדוגמה MDITest מכיל בשגרת Initialize של טופס הצאצא, קוד אשר מגדיל ערך משתנה ציבורי כדי לדעת כמה טפסי צאצא נוצרו ולאחר מכן קובע את מאפיין Caption, של הטופס החדש, כך שיציג את ערך המשתנה.

שימוש בתפריטים

בפרק 6 **שליטה נוספת למשתמש: תפריטים וסרגלי כלים**, למדת כיצד ליצור תפריט לתוכנית. בנוסף, גילית שתוכל ליצור תפריט נפרד עבור כל טופס בתוכנית אם תרצה בכך. טפסי MDI יכולים גם הם להכיל תפריטים. תוכל ליצור תפריט עבור טופס MDI באותה דרך שיצרת תפריט עבור טופס רגיל, תוך שימוש בעורך התפריטים. תפריט השייך לטופס MDI הוא הדרך העיקרית לגישה לאפשרויות הקיימות ביישום MDI.

ראה: "יצירת שורת התפריטים", פרק 6.

ביישום MDI טפסי צאצא יכולים להכיל גם הם תפריטים. יצירת תפריטים בטפסי צאצא דומה ליצירת תפריטים בטופס אב, ונעשית באמצעות עורך התפריטים (Menu Editor). אולם, כאשר מופיע על המסך טופס צאצא המכיל תפריט, יופיע התפריט כחלק משורת התפריט של טופס האב ולא כחלק מטופס הצאצא. תופעה זו יוצרת בעייתיות מסוימת ביישומי MDI, הנובעת מכך שכאשר טופס הצאצא פעיל, התפריטים שלו מחליפים למעשה את תפריטי טופס האב. כלומר, כאשר טופס צאצא עם תפריטים נמצא במיקוד, לא ניתן לגשת לתפריטי טופס האב.

תוכל להשתמש בשתי דרכים לפתרון בעיות הנובעות מהחלפת התפריטים בין טופס הצאצא וטופס האב. הדרך הראשונה היא להעתיק את כל הפונקציות הנחוצות מטופס האב לכל אחד מהטפסים הצאצאים. לרוע המזל, גישה זו עלולה להוביל ליצירת תוכנית מסורבלת וקשה לניהול, בעיקר אם קיימים טפסי צאצא רבים המכילים תפריטים.

דרך חלופית לפתרון הבעיה היא לכלול את כל תפריטי הטפסים הצאצאים בתפריט האב. במצב זה תוכל לכתוב קוד בשגרות GotFocus ו-LostFocus של טופס הצאצא, כדי להציג את תפריטי האב אשר מיועדים לטופס הצאצא הנמצא במוקד, ולהסתיר את תפריטי הטפסים אשר אינם במיקוד. לדוגמה, כשאתה עובד על תוכנית עיבוד תמלילים, תרצה שתפריטי **קובץ** (File) **ועזרה** (Help) יהיו זמינים במשך כל זמן פעולת התוכנית, לעומתם, תפריטי **עריכה** (Edit) **ועיצוב** (Design) צריכים להיות זמינים רק בעת עבודה על מסמך. כדי להפוך את התפריטים **עריכה ועיצוב** לגלויים, תוכל לכתוב קוד כגון הקוד המוצג בהמשך בשגרת GotFocus של טופס הצאצא המתאים:

```
mnuEdit.Visible = True
mnuFormat.Visible = True
```

הקוד הבא המוצג בשגרת LostFocus של טופס הצאצא, הופך בחזרה את תפריטי **עריכה ועיצוב** למוסתרים:

```
mnuEdit.Visible = False
mnuFormat.Visible = False
```

כדי לוודא שכל קוד תפריט בטופס האב פועל על טופס הצאצא המתאים, השתמש במאפיין ActiveForm של טופס האב כמתואר בסעיף הקודם.

טיפול בטפסי צאצא

אחד מיתרונות העבודה עם יישומי MDI היא היכולת לטפל בטפסי צאצא בקלות. Visual Basic מציעה מספר כלים אשר מקלים את העבודה עם מספר טפסי צאצא הפתוחים על המסך בעת ובעונה אחת. תוכל לצייד את המשתמש ביכולת לסדר את חלונות הצאצאים באופן אוטומטי. תוכל אפילו ליצור תפריט המכיל את רשימת הטפסים הפעילים ולאפשר למשתמש גישה לכל אחד מהם על ידי בחירת הטופס הרצוי מתוך התפריט. אפשרויות אלו שימושיות בעיקר כאשר המשתמש מבצע מעבר שוטף בין משימות או בין קבצים ביישום.

שימוש בסידור אוטומטי

דרך אחת אשר בעזרתה המשתמש יכול לעבור בין קבצים פתוחים היא על ידי הצגת כל טופס על המסך בסדר מסוים. בדרך זו, המשתמש יכול לגשת לכל טופס בלחיצת עכבר. הבסיס לכך היא שיטת Arrange של טופס MDI. שיטה זו מסדרת את כל הטפסים הצאצאים של היישום בסגנון מסוים. כל אחת מצורות העיצוב הללו גורמת לכך שלפחות חלק מכל טופס יהיה גלוי לעיני המשתמש. השימוש בשיטה זו נפוץ בתוכניות MDI מבוססות Windows.

כדי להשתמש בפקודת Arrange, עליך לציין את שם טופס MDI, השיטה עצמה וקבוע המייצג את צורת הסידור שברצונך לבצע על הטפסים. שורת הקוד הבאה מדגימה את השימוש בשיטה:

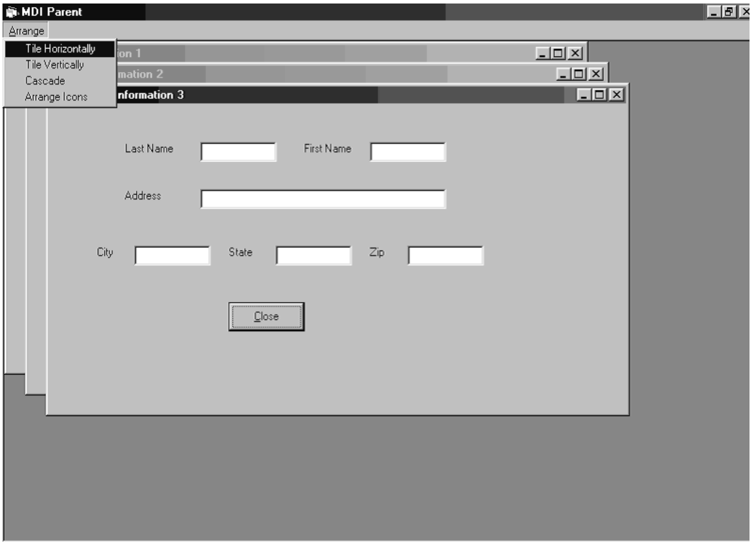
```
mdiMain.Arrange vbCascade
```

תוכל לסדר את החלונות בארבע צורות אפשריות בעזרת שיטת Arrange. כל אחת מהצורות מיוצגת על ידי קבוע. טבלה 17.1 מכילה סיכום של כל צורות הסידור. דוגמה לצורות הסידור ניתן לראות בתרשימים 17.12 עד 17.15.

טבלה 17.1: צורות סידור לטפסי צאצא

קבוע	תיאור
vbCascade	כל טפסי הצאצא הפתוחים מסודרים כך שכל טופס מונח מעל הטופס הקודם עם סטייה מסוימת
vbTileVertical	גובה טפסי הצאצא הפתוחים זהה לגובה המירבי של טופס האב והם מוצגים אחד לצד השני. אם קיימים טפסי צאצא רבים, הם יוכלו לתפוס מספר שורות כאשר הם מסודרים
vbTileHorizontal	רוחב טפסי הצאצא הפתוחים זהה לרוחב טופס האב והם מונחים אחד על גבי השני. אם יש טפסי צאצא רבים, הם עלולים לתפוס מספר עמודות
vbArrangeIcons	סמלים ממוזערים של טפסי צאצא מסודרים בתחתית טופס האב

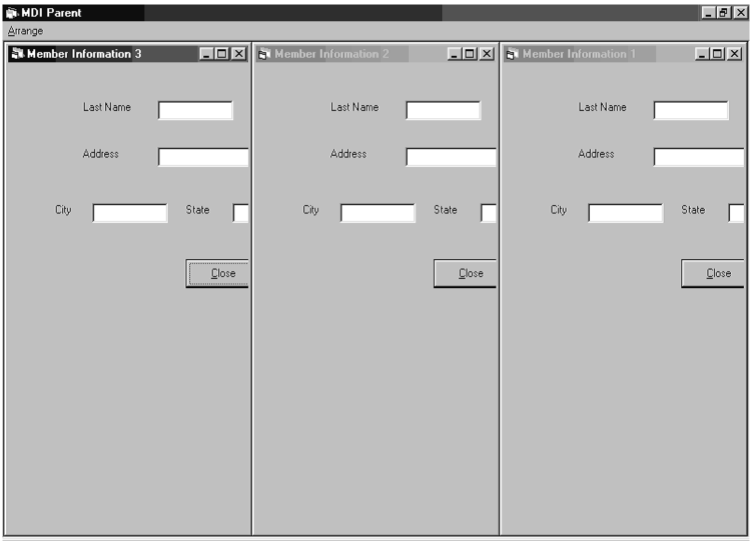
לרוב תרצה להציב את אפשרויות הסידור בתפריט **Window** בטופס MDI. כל אפשרות סידור שתרצה לתמוך בה תופיע כפריט נפרד בתפריט.



תרשים 17.12: טפסי הצאצא מסודרים אחד מעל לשני

טיפול ברשימת חלונות

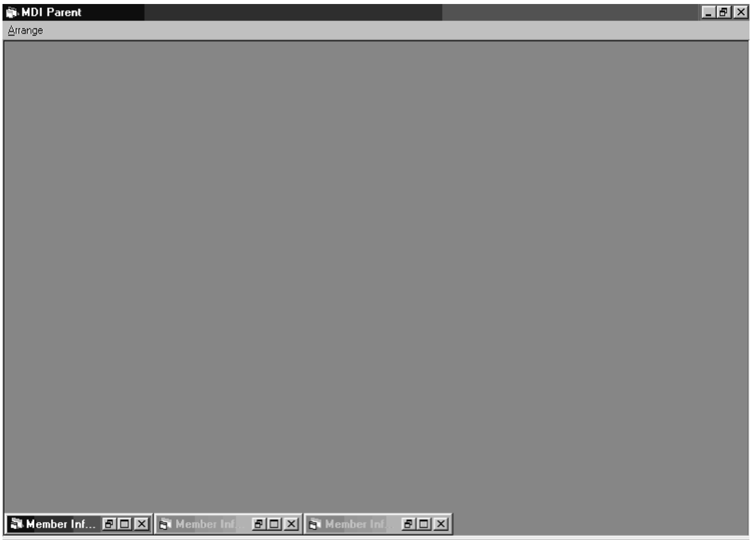
דרך אחרת ליצור גישה קלה לטפסי צאצא ביישום, היא להחזיק רשימת קבצים פתוחים. מלאכה זו קלה למדי. תוכל ליצור את רשימת החלונות בעת יצירת התפריטים עבור טופס האב. עליך לקבוע באיזה תפריט לשבץ את הרשימה ולאחר מכן להגדיר את ערך המאפיין WindowList בעורך התפריטים, כ-True. פעולה זו מתוארת בתרשים 17.16.



תרשים 17.13: טפסי צאצא אשר סודרו בצורה אנכית



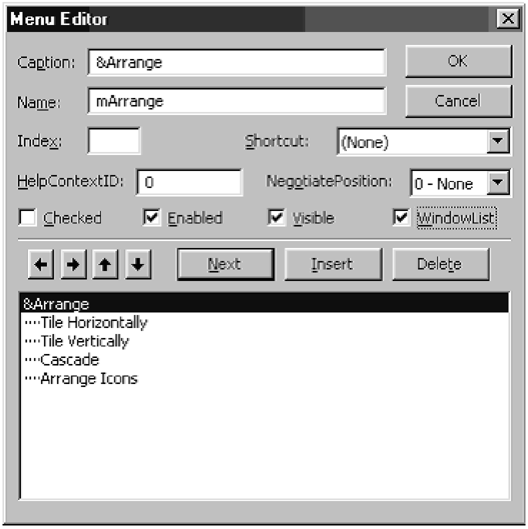
תרשים 17.14: טפסי צאצא אשר סודרו בצורה אופקית



תרשים 17.15: כאשר טפסי צאצא ממוזערים, הם מוצגים **כסמלים** המסודרים בתחתית טופס האב

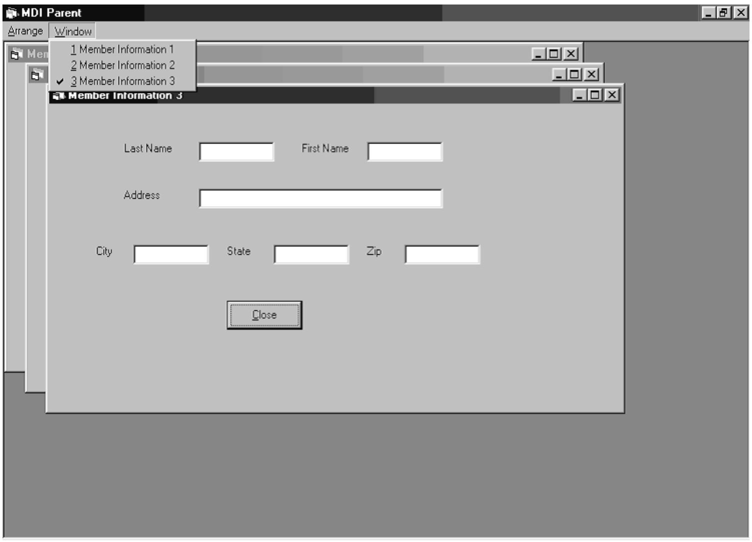
הערה:

תוכל גם לשנות את הגדרות מאפיין WindowList מתוך קוד התוכנית.



תרשים 17.16: חלונות הצאצאים פתוחים

בעת הוספת טפסי צאצא ליישום, רשימת החלונות מתעדכנת באופן אוטומטי ומציגה את הטפסים החדשים שהתווספו. שם הפריט המשויך לטופס הוא הטקסט שניתן לטופס שיצרת. לשם הטופס הפעיל הנוכחי מצורף הסימן (✓). תרשים 17.17 מציג רשימת חלונות עבור יישום MDI.



תרשים 17.17: רשימת החלונות מאפשרת למשתמש לבחור את הטופס עמו יעבוד

תוכנית דוגמה - יישום MDI לניהול קשרים

כרגיל, הדרך הטובה ביותר להדגים את טכניקת יצירת יישומי MDI היא לבנות יישום שימושי. התוכנית מכילה מספר מופעים של טופס מקור ולכן היא הופכת ליישום MDI.

אם חלק מעבודתך כולל קשר עם לקוחות קבועים, אתה בוודאי משתמש בתוכנה כלשהי לניהול הקשרים. סוג זה של תוכניות מאפשר לך לשמור מידע על כל אחד מהלקוחות כגון שמם, כתובתם, מספר הטלפון שלהם, תאריך אחרון בו נוצר קשר וכיוצא בזה. בחלק מתוכניות אלו קיימת מגבלה של אפשרות לעבוד בכל פעם עם פרטי לקוח אחד בלבד. מגבלה זו עלולה ליצור אי נוחות, למשל במצב בו אתה עובד על הזמנה עבור לקוח מסוים, ובאמצע העבודה מתקשר לקוח אחר כדי לשוחח אתך על ההזמנה שלו. במקרה מעין זה, עליך לסגור את חלון המידע עבור הלקוח שעבדת עליו ולפתוח את חלון המידע עבור הלקוח השני. תאר לך שתוכל לפתוח את חלון המידע עבור הלקוח השני ללא צורך בסגירת החלון הראשון. יכולת מעין זו היתה מקלה עליך את העבודה עד מאוד הלא כן? ובכן, תוכל לעשות זאת בעזרת יישום MDI.

סעיף זה ילמדך כיצד לבנות תוכנית MDI מעין זו. התוכנית תציג רק את שם וכתובת הלקוח ותשמש מעין אב-טיפוס לתוכניות מורכבות יותר שתוכל לפתח בעצמך. כדי ליצור תוכנית מלאה לניהול קשרים, עליך להוסיף קוד נוסף לטיפול במסד הנתונים. התוכנית משתמשת במסד הנתונים Microsoft Access ובמנגנון Jet לקבלת המידע.

ראה: "שימוש ב-Table", פרק 26.

יצירת טופס MDI

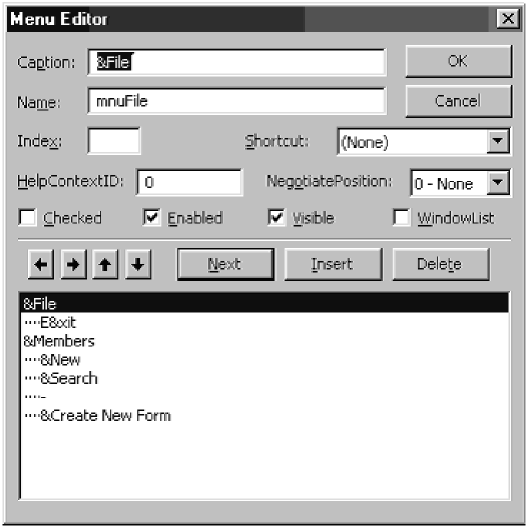
השתמש בידע שרכשת בסעיפים קודמים והוסף טופס MDI לתוכנית. אחר, קבע את מאפיין AutoShowChildren של הטופס לערך False. כן עליך לשנות את מאפייני Name ו-Caption של הטופס כך שיכילו ערכים שונים מערכי ברירת המחדל.

בתום קביעת מאפייני הטופס, עליך ליצור תפריט אשר יציג את המידע הקשור ללקוח בטופס צאצא מתאים. פריטי התפריט שעליך להוסיף מוצגים בתרשים 17.18.

ראה: "יצירת שורת תפריטים", פרק 6.

פריט אחד מתוך התפריט אשר ראוי לציון מיוחד הוא Create New Form. המשתמש יכול לבחור באפשרות זו כדי לקבוע האם להציג את המידע בחלון הצאצא הנוכחי או ליצור חלון חדש בכל פעם. מאפיין Checked של הפריט נקבע בהתאם להחלטת המשתמש.

לאחר שיצרת את התפריט, עליך כמובן, להוסיף קוד אשר יפעיל את האפשרויות בתפריט. תוכנית 17.2 מכילה את הקוד עבור הפריטים המוצגים בתרשים 17.18.



תרשים 17.18: תוכנית הדוגמה לניהול הקשרים מכילה את פריטי התפריט המוצגים

תוכנית 17.2:mdiMain.frm - שימוש בקוד בתפריט היישום לניהול קשרים.

```
Private Sub filExit_Click()
        Unload Me
End Sub

Private Sub MDIForm_Load()
        Me.WindowState = vbMAximized
End Sub

Private Sub MDIForm_Unload(Cancel As Integer)
        CustDb.Close
End Sub

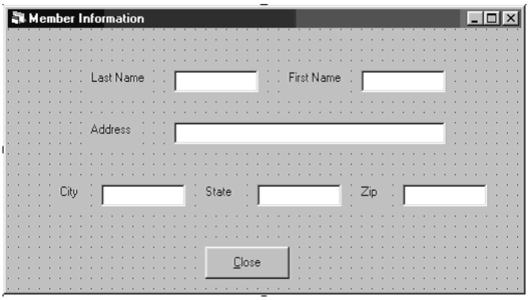
Private Sub memCreate_Click()
Dim CheckSet As Boolean
        CheckSet = Not memCreate.Checked
        memCreate.Checked = CheckSet
        CreateForm = CheckSet
End Sub

Private Sub memNew_Click()
        If CreateForm Then
                Dim frmMem As New frmMember
                frmMem.Show
        End If
        ClearCust
End Sub
```

```
Private Sub memSearch_Click()
	frmSearch.Show vbModal
	If CreateForm Then
		Dim frmMem As New frmMember
		frmMem.Show
	End If
	ShowCust
End Sub
```

יצירת טופס צאצא ללקוח

בשלב הבא עליך ליצור את טופס הצאצא בו בו יוצג המידע אודות הלקוח. כדי ליצור את הטופס, הוסף טופס חדש לפרויקט (או השתמש בטופס ההתחלתי) וקבע את מאפיין MDIChild שלו לערך True. בנוסף, יהיה עליך לשנות את ערכי המאפיינים Name ו-Caption (הגדר את שם הטופס frmMember כדי להתאימו לקוד פריטי התפריט). לאחר קביעת מאפייני הטופס, הוסף את הפקדים הנחוצים להצגת המידע. הטופס המלא מוצג בתרשים 17.19.



תרשים 17.19: המידע אודות הלקוח מוצג בטופס צאצא

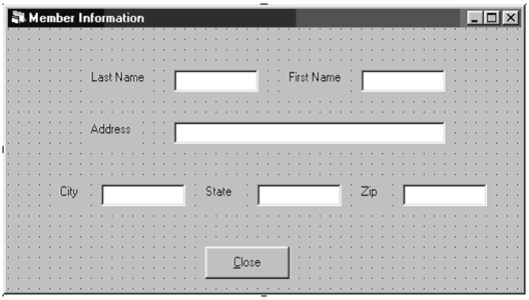
יצירת טופס חיפוש

כאשר אתה מעוניין לאתר מידע הנוגע ללקוח מסוים מתוך מאגר הלקוחות, עליך להשתמש בטופס חיפוש המאפשר לך להקליד שם שיש לחפשו. קרא לטופס החיפוש בשם frmSearch. על הטופס להיות פשוט ביותר, לכלול תווית, תיבת טקסט לתוכה יוקלד שם הלקוח, ושני לחצני פקודה בעזרתם ניתן לאשר או לבטל את פעולת החיפוש. הקוד עבור הטופס פשוט גם הוא. הוא ישתמש בשיטת FindFirst של מערך הרשומות הרלוונטי כדי לאתר את המופע הראשון של השם המתאים לפקודת החיפוש. טופס החיפוש המלא מוצג בתרשים 17.20, והקוד עבור הטופס מוצג בתוכנית 17.3.

הערה:
כדי שהתוכנית תעבוד, הוסף הפניה (Reference) ל-DAO (ראה פרק 11). בחר בתפריט Project, Reference וסמן Microsoft DAO 3.51 Object Library. כמו כן עליך לבדוק שמסד הנתונים המצוין בתוכנית נמצא במקום המתאים.

הערה:

מערך רשומות (RecordSet) הוא סוג מיוחד של אובייקט כאשר משמש כקשר בין תוכנית של Visual Basic למידע הנמצא בבסיס נתונים. תלמד על מערכי רשומות בפרק 26 **שימוש באובייקטי גישה לנתונים (DAO)**.



תרשים 17.20: שכלל את טופס החיפוש על ידי הוספת אפשרות לחיפוש על פי שם פרטי

תוכנית 17.3: MDISearch.txt - השיטה FindFirst לאיתור הלקוח המבוקש.

```
Private Sub cmdCancel_Click()
      Unload Me
End Sub

Private Sub cmdSearch_Click()
      Dim SrchStr As String
      SrchStr = txtSearch.Text
      CustRset.FindFirst "LastName = '" & SrchStr & "'"
      Unload Me
End Sub
```

לב התוכנית

הטפסים שיצרת עד כה מהווים את ממשק התוכנית, אולם לב התוכנית הוא קבוצת שגרות האחראית על הצגת המידע הלכה למעשה. כדי ליצור אותן, עליך להוסיף מודול (Module) לתוכנית. תוכל לעשות זאת על ידי בחירת **Add** מתפריט **Project** או על ידי בחירת **Module** מתוך התפריט הנפתח לאחר לחיצה על הלחצן **Add Object**.

ראה: "הגדרת טווח ההכרה של שגרות ופונקציות", פרק 11.

לאחר הוספת מודול לפרויקט, עליך להגדיר מספר משתנים ציבוריים וליצור את השיגרה אשר תגדיר את התוכנית. משתנים ציבוריים יוצרים עבור התוכנית נגישות לאובייקט מסד הנתונים. בתום קביעת המשתנים, עליך ליצור שגרת Sub Main לשם אתחול המידע במסד הנתונים והצגת טופס האב. בתוכנית 17.4 ניתן לראות את ההצהרה על המשתנה הציבורי ואת השיגרה Sub Main.

תוכנית 17.4: MDIChild.txt - שימוש בשיגרה Sub Main כדי לאתחל את מסד הנתונים ולטעון את הטופס הראשי

```
Public CustDb As Database, CustRset As RecordSet
Public CreateForm As Boolean

Sub Main()
    Set CustDb = DBEngine.Workspaces(0).OpenDatabase("D:\VB6Book\NewDb.mdb")
    Set CustRset = CustDb.OpenRecordset("Customers", dbOpenDynaset)
    mdiMain.Show
    CreateForm = True
End Sub
```

לאחר יצירת שגרת Sub Main, עליך לשנות את אפשרויות הפרויקט ולקבוע את Sub Main כאובייקט ההתחלתי של התוכנית.

בתוכנית 17.5 תמצא את שתי השגרות ClearCust ו-ShowCust אשר מציגות מידע אודות לקוח קיים או מעלות עבורך את טופס המידע לקליטת לקוח חדש. שגרות אלו נקראות על ידי פריטי התפריט המתאימים בטופס MDI. ראוי לציין כי בשגרות אלו נעשה שימוש במאפיין ActiveForm של טופס האב כדי לקבוע באיזה מבין טפסי הצאצא יתקבלו הנתונים שיישלחו.

תוכנית 17.5: MDIChild.txt **-** שימוש במאפיין ActiveForm כדי לשלוח את פלט השיגרה למקום הנכון.

```
Public Sub ClearCust()
Dim I As Integer
      For I = 0 To 5
            mdiMain.ActiveForm.txtMember(I).Text = ""
      Next I
End Sub

Public Sub ShowCust()
      With mdiMain.ActiveForm
            .txtMember(0).Text = CustRset!LastName & ""
            .txtMember(1).Text = CustRset!FirstName & ""
            .txtMember(2).Text = CustRset!Address1 & ""
            .txtMember(3).Text = CustRset!City & ""
            .txtMember(4).Text = CustRset!State & ""
            .txtMember(5).Text = CustRset!Zip & ""
      End With
End Sub
```

הרצת התוכנית

כאשר תריץ את התוכנית, תוכל ליצור חלונות חדשים עבור כל לקוח שתוסיף, או לשנות את מצב האפשרות Create New Form בתפריט כדי להציג את המידע על הלקוח הבא באותו החלון. כפי שכבר צוין, תוכנית זו מיועדת להדגים את הרעיון באופן כללי ואתה מוזמן לצרף אליה מרכיבים משלך. התוכנית מוצגת בתרשים 17.21.



תרשים 17.21: תוכל להציג מידע אודות מספר לקוחות בעת ובעונה אחת

שיפור היישום

פרק זה מדגים מספר טכניקות בהן תוכל להשתמש כדי ליצור יישומי MDI. כפי שאתה רואה, טופס MDI יכול להוות כלי רב עוצמה בעזרתו ניתן ליצור תוכניות. אולם, יש מספר נקודות שעליך להביא בחשבון כדי שתוכניותיך תוכלנה לתפקד בצורה אופטימלית. הקפדה על הנקודות המובאות להלן תסייע לך לבנות תוכניות בעלות רמת ביצועים גבוהה ומינימום בעיות למשתמש :

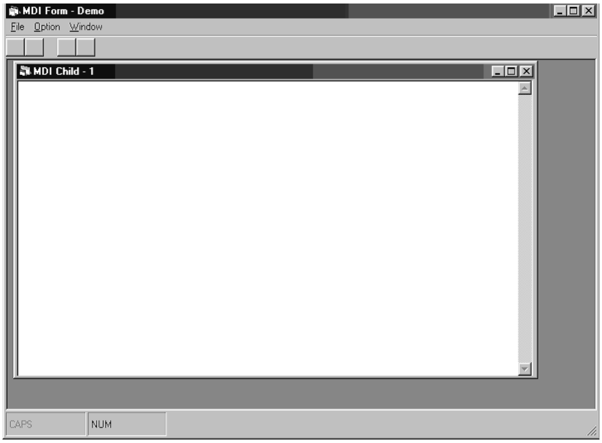
- כל חלון צאצא המוצג על המסך צורך זיכרון. כאשר אתה טוען חלונות רבים מסוג זה, היישום צורך כמות זיכרון רבה ולכן עליך ליצור חלונות צאצאים המכילים כמות מינימלית של קוד ופקדים, או לחילופין, להגביל את כמות החלונות הפתוחים.
- אם תפריטי טופס הצאצא וטופס האב מכילים את אותן האפשרויות, כתוב את הקוד המיועד להן בשגרות טופס האב. בצורה זו, שגרות Click של תפריטי הצאצאים יקראו לשגרת Click של טופס האב לצורך שימוש בקוד משותף.
- שנה את מצב שגרות Click של טפסי האב והצאצאים מ-Private ל-Public זאת כדי לאפשר שימוש בשגרות משותפות לחלון האב וחלונות הצאצאים.
- הימנע משימוש במאפיין Name בקריאה לטופס הצאצא. במקום זאת, השתמש בקבוע Me (או הימנע מזיהוי טופס). כמו כן השתמש בפונקציה ActiveForm בשגרות טופס האב.

- הצב את **כל** הפקדים המוסתרים (Common Dialog Control, פקדי Image List וכיוצא בזה) בטופס האב. דבר זה מאפשר לכל טפסים הצאצא להשתמש בפקדים אלה ללא צריכת זיכרון מיותר.

שמירה על כללים אלה תביא ליצירת קוד פשוט יותר, ותשפר את ביצועי היישום.

יצירת מסגרת בסיסית ליישום MDI

הקוד המתואר בסעיף זה מתוכנן כך שיספק שלד בסיסי לכל יישום MDI שתיצור בעתיד. ניתן לשנות את הקוד הנמצא בשלד זה כדי להתאים אותו לדרישות הספציפיות של כל יישום. לאחר מכן תוכל להשתמש בפרויקט השלד כמקור ליישומים אחרים. היישום המלא מוצג בתרשים 17.22.



תרשים 17.22: ניתן לפשט את העבודה על יישומי MDI על ידי יצירת יישום מקור לדוגמה

יצירת תבנית טופס אב MDI

טופס אב הוא למעשה מחסן של חלונות הצאצאים ולפיכך הוא אחראי על יצירת חלונות חדשים כאלה. לפיכך, לרוב מטילים על טופס האב גם את המשימה להשגיח על מספר טפסי הצאצא שנוצרו על ידו. בנוסף, טופס אב מכיל בדרך כלל אלמנטים משותפים השייכים לממשק המשתמש כמו סרגל כלים, שורת המצב וכדומה.

תוכנית 17.6 מציגה דרך בה משגיחים על מספר החלונות. שגרות WindowCreated ו-WindowDestroyed נקראות על ידי טפסי הצאצא בשגרות Form_Load ו-Form_Unload שלהם בהתאם. משתנה ChildWindowCount הוא משתנה ציבורי (Public) המאפשר לטפסי הצאצא לדעת כמה טפסים מסוגם מצויים בזיכרון.

תוכנית 17.6: MDIAPP.VBP טופס אב המכיל קוד משותף לכל טפסי הצאצא.

```
'************************************************************
' MDIParent.from – Demonstrates some basic concepts on how a MDI parent
'  form should behave in an MDI application.
'************************************************************
Option Explicit
Private mintChildWinCount As Integer
'************************************************************
' Returns how many child windows have been created
'************************************************************
Public Property Get ChildWindowCount() As Integer
      ChildWindowCount = mintChildWinCount
End Property
'************************************************************
' Called when a window is created to increment the window counter
'************************************************************
Public Sub WindowCreated()
      MintChildWinCount = mintChildWinCount + 1
      UpdateButtons True
End Sub
'************************************************************
' Called when a window is created to decrement the window counter
'************************************************************
Public Sub WindowDestroyed()
      MintChildWinCount = mintChildWinCount – 1
      UpdateButtons minChildWinCount
End Sub
```

הערה:
שגרת UpdateButtons הינה בגדר המלצה **ואינה** מיושמת בדוגמה.

שים לב כי התוכנית לעיל מכילה קריאה ל-UpdateButtons. שיגרה זו מפעילה ומנטרלת את הלחצנים בסרגל הכלים. אם קיים טופס צאצא, לחצני סרגל הכלים מופעלים. כאשר טופס הצאצא האחרון נמחק מהזיכרון, שגרת WindowDestroyed מפחיתה 1 מערך המשתנה mintChildWinCount עד שערכו מגיע ל-0 וגורמת במצב זה לחסימת הלחצנים שעל סרגל הכלים.

הקטע החשוב ביותר בתוכנית MDIPARENT.FRM הוא שגרת Click של תפריט **File**. קוד זה אחראי ליצירת חלונות, פתיחת קבצים ויציאה מהיישום.

בגלל העובדה שכל הפעולות בתפריט **File** קיימות בטופס האב ובטפסי הצאצא, עליך להפוך שיגרה זו לציבורית (Public, ראה תוכנית 17.7).

תוכנית 17.7: MDIAPP.VBP טיפול בשגרת Click של התפריט.

```
'*************************************************************
' File menu handler for the MDI form when no windows are displayed.
' In this demo the child windows will have a menu just like this,
' so we will make this public so the children can call this event.
'*************************************************************
Public Sub mnuFileItems_Click(Index As Integer)
	Select Case Index
	'*********************************************************
	Case 1
		Dim frmNew As New frmChild
		frmNew.Visible = True
	'*********************************************************
	' File Open – Prompt the user for a filename, then load
	' it into the child window (in OpenFile) if the user didn't
	' press cancel in the dialog.
	'*********************************************************
	Case 2
		On Error Resume Next
		With cdlg
			.Flags = cdlOFNFileMustExist
			.Filter = "Text Files (*.txt) |.txt | All Files (*.*) | *.*"
			.ShowOpen
		End With
		If Err <> cdlg.cdlCancel Then OpenFile cdlg.filename
	'*********************************************************
	' Index 3 is the seperator, so don't do anything
	'*********************************************************
	Case 3
	'*********************************************************
	' File Exit – Terminate the application
	'*********************************************************
	Case 4
		Unload Me
	End Select
End Sub
```

כאשר נבחרת אפשרות **New** בתפריט **File** (Index = 1), השורה Dim frmNew As New frmChild יוצרת מופע חדש של טופס הצאצא. אולם הפעולה אינה יוצרת את הטופס החדש בעצמה. הטופס נוצר ברגע שהקוד פונה לאחד ממאפייניו או שגרותיו. לכן, השורה frmNew.Visible = True היא זו היוצרת את הקוד בסופו של דבר. לאחר שהטופס נוצר, למאפיין Visible שלו נקבע ערך True והטופס מוצג על המסך.

טיפ:

טפסים הנוצרים על ידי פקודת New אינם מוגדרים על ידי ברירת המחדל כך שיוצגו ולפיכך עליך לזכור להציגם על ידי קביעת ערך מאפיין Visible שלהם לערך True.

קוד האפשרות **File, Open** (Index = 2) בתוכנית 17.8 מציג למעשה תיבת דו-שיח כדי שהמשתמש יספק שם קובץ. אם המשתמש לא לוחץ על Cancel, הרי שהקובץ נפתח בעזרת שגרת OpenFile, כמו שמוצג בתוכנית 17.8. הפריט האחרון בפקודה הנבחרת הוא Index 4, אשר מייצג את האפשרות **File, Exit** בתפריט. קוד זה יהיה קל לכתיבה מפני שהדרך הנכונה לסגור יישום MDI היא להסיר את טופס MDI מהזיכרון (בעזרת שגרת Unload).

כמו שצוין קודם לכן, הקוד בשגרת OpenFile אחראי על פתיחת קובץ טקסט וטעינתו לתוך תיבת טקסט בטופס הצאצא. קוד זה נכתב בצורתו הבסיסית בלבד והוא אינו מכיל טיפול בשגיאות כגון בדיקה אם גודל הקובץ עולה על 44K תחת Windows 95. אולם, הוא עדיין מספק דוגמה בסיסית כיצד לטעון קובץ אל תוך תיבת טקסט, אשר תספיק לצורך הדגמה זו.

אזהרה:

הימנע משימוש בפקודת End כדי לסגור את היישום שלך. פקודת End מפסיקה את פעולת היישום באופן מיידי ולא מאפשרת לשגרת Form_Unload להתבצע. הדרך הטובה ביותר לסיים את פעולת יישום MDI היא להסיר (**Unload**) את הטופס.

תוכנית 17.8: MDIAPP.VBP קוד משותף.

```
'*********************************************************
' Code shared among the child windows should be put in either a
' module or the MDI parent form. This OpenFile code will be used
' by all of the children, so we will keep it in the MDI parent form.
'*********************************************************
Public Sub OpenFile(strFileName As String)
    Dim strFileContents As String
    Dim intFileNum As Integer
    '*********************************************************
    ' Get a free file handle
    '*********************************************************
    intFileNum = FreeFile
    '*********************************************************
    ' Open the file
    '*********************************************************
```

```
    Open strFileName For Input As intFileNum
        '*****************************************************
        ' Put the contents of the file into the txtData control of
        ' the child form. This code will fail if the file is too
        ' large to fit in the textbox, so you should include
        ' additional error handling in your own code.
        '*****************************************************
        With ActiveForm
            .txtData.Text = Input$(LOF(intFileNum), intFileNum)
            '*************************************************
            ' Set the caption of the child form to the filename
            '*************************************************
            .Caption = strFileName
        End With
    '*********************************************************
    ' Always close files you open as soon as you are don't with them
    '*********************************************************
    Close intFileNum
End Sub
```

בוודאי שמת לב כי פעולתה של שגרת OpenFile מסתכמת בטעינת הקובץ אל תוך תיבת הטקסט בחלון הפעיל על ידי שימוש במאפיין ActiveForm. ניתן להשתמש בשיטה זו מכיון שהתפריט הפעיל תמיד יהיה התפריט השייך לטופס הפעיל. תוכל להניח שכל פעולה אשר תתבצע בשגרות התפריט תתיחס לטופס הפעיל מפני שהדרך היחידה לפתוח קובץ היא דרך התפריט.

טופס הצאצא

כעת, לאחר שהבנת מהו תחום האחריות של טופס האב, הבה ונראה מה תפקידו של טופס הצאצא בסדר הפעולות.

כפי שצוין קודם לכן, טפסי צאצא אחראים על קריאה לשגרות WindowCreated ו-WindowDestroyed של טופס האב. תוכנית 17.9 מדגימה כיצד נעשית פעולה זו מתוך שגרות Form_Load ו-Form_Unload. בנוסף, טופס הצאצא קובע את הכותרת ההתחלתית שלו לפי המאפיין ChildWindowCount של טופס האב. למרות ששיטה זו טובה לדוגמה שהובאה כאן, תוכל לכתוב קוד אשר יהפוך את כותרת הטופס למורכבת יותר. מה יקרה לדעתך במצב בו כאשר פתוחים על המסך שלושה חלונות, תסגור את החלון השני ותיצור חלון נוסף? כיצד ניתן להימנע מהבעיה?

תוכנית 17.9: MDIAPP.VBP שימוש בטופס הצאצא כדי לכתוב קוד מסוים עבור כל צאצא.

```
'************************************************************
' MDICHILD.frm – Demonstrates some basic techniques on how a MDI child
' window should behave.
'************************************************************
Option Explicit
'************************************************************
' When a new form is created, it should call the WindwCreated function
' in the MDI parent form (which increments the window count in this
' case). It should also set its caption to distinguish it from other
' child windows.
'************************************************************
Private SubForm_Load()
	MDIParent.WindowCreated
	'************************************************************
	' This works, but it has a fatal flaw.
	'************************************************************
	Caption = Caption & " – " & MDIParent.ChildWindowCount
End Sub
'************************************************************
' Make sure txtData always fills the client area of the form.
'************************************************************
Private Sub Form_Resize()
	TxtData.Move 0, 0, ScaleWidth, ScaleHeight
End Sub
'************************************************************
' Let the MDI parent know that this window is being destroyed.
'************************************************************
Private Sub Form_Unload(Cancel As Integer)
	MDIParent.WindowDestroyed
End Sub
```

פרט קטן נוסף שבוודאי הבחנת בו בקוד זה הוא שגרת Form_Resize. שיגרה זו מוודאת שפקדי תיבת הטקסט מכסים תמיד את השטח המוקצה לפרטי הלקוח בטופס. זכור ששיגרה זו פועלת על פקד מכל סוג שהוא.

תפיסה נוספת אשר הוצגה קודם לכן היא העובדה שעל טפסי הצאצא להשתמש במידת האפשר, בשמות שגרות של תפריט האב. תוכנית 17.10 מכילה את הקוד המטפל בכל התפריטים של תוכנית MDICHILD.FRM.

תוכנית **17.10**: MDIAPP.VBP טיפול בתפריט היישום.

```
'**************************************************************
' Since the child File menu is identical to the MDI parent file menu,
' we should avoid duplicate code bu calling the parent's mnuFileItems
' click Event.
'**************************************************************
Private Sub mnuFileItems_Click(Index As Integer)
	MDIParent.mnuFileItems_Click Index
End Sub
'**************************************************************
' The options menu is unique to the child forms, so the code should
' be in the child form or separate BAS module.
'**************************************************************
Public Sub mnuOptionsItems_Click(Index As Integer)
	'**********************************************************
	' Don't stop for errors.
	'**********************************************************
	On Error Resume Next
	'**********************************************************
	' Show the color dialog (since all menu items here need it)
	'**********************************************************
	MDIParent.cdlg.ShowColor
	'**********************************************************
	' If the use selected cancel, then exit
	'**********************************************************
	If Err = cdlCancel Then Exit Sub
	'**********************************************************
	' Otherwise set the color based on the value returned from the dlg
	'**********************************************************
	Select Case Index
		Case 1 'Backcolor…
			TxtData.BackColor = MDIParent.cdlg.Color
		Case 2 'Forecolor
			TxtData.ForeColor = MDIParent.cdlg.Color
	End Select
End sub
'**************************************************************
' If you set your indexes of your window menu properly, you can save
' yourself some code. I was careful to make sure my window menu items
' indices were equivalent to the possible values for the Arrange method.
'**************************************************************
Private Sub mnuWindowItems_Click(Index As Integer)
	MDIParent.Arrange Index
End Sub
```

התפריט הראשון הוא תפריט **File**. תפריט זה זהה לתפריט הראשון בטופס האב ולכן תוכל פשוט לקרוא לשגרת mnuFileItems_Click בטופס האב כדי לבצע את הפעולות הרגילות בתפריט. התפריט השני הוא תפריט Options אשר מופיע בטופס הצאצא בלבד. לתפריט זה תוכל לכתוב קוד בשגרות טופס הצאצא. הפוך את השגרות הללו לציבוריות (Public) כדי לאפשר לפקדי סרגל הכלים הנמצאים בטופס האב גישה אליהן. בנוסף, השתמש בפקד CommonDialog הנמצא בטופס האב כדי להציג את תיבת הדו-שיח **Color**.

טיפ:

אם אחד או יותר מפריטי התפריט בטופס הצאצא דורש מעל ל-12 שורות קוד (בלי להתחשב בהצהרות Dim, הערות ושורות רווחים), עליך להעביר את הקוד למודול (Module) משותף או לטופס האב. דבר זה מונע מהקוד לצרוך יותר מדי זיכרון פנוי בכל פעם שנוסף טופס חדש.

לבסוף, קיים תפריט Window אשר מתייחס לטפסי הצאצא בלבד (למרות שמתפקידו של טופס האב לטפל בתפריט זה). על ידי תכנון מערך פקדי תפריטים בצורה נבונה, תוכל לכתוב קוד לתפריט זה אשר יכיל שורת קוד בודדת.

הערה:

הקוד המלא נמצא בשני קבצים: MDIChild.txt ו-MDIParent.txt, בתיקיה MDISkeleton.

טיפ:

תוכל להוריד פרויקט נוסף מאתר Macmillan אשר עושה שימוש מלא בכל השיטות לתכנות בסביבת MDI שנלמדו בפרק זה. זהו עורך טקסט MDI הנקרא "MDI text editor". יישום זה דומה לפנקס הרשימות של Windows, למעט העובדה שהוא מאפשר פתיחה של מספר מסמכים בו-זמנית. תוכל לפתוח את הפרויקט מתוך Visual Basic. הקוד מכיל הערות המסבירות את אופן פעולתו.

מכאן...

פרק זה היווה מבוא ליצירת יישומי MDI בעזרת Visual Basic. תוכל לקבל מידע נוסף אודות חלק מהנושאים אשר נדונו בפרק זה, בכל אחד מן הפרקים הבאים:

- כדי לרענן את ידיעותיך בנושא יצירת טפסים, עיין בפרק 3, **אבני היסוד של Visual Basic**.
- כדי להעמיק את היכרותך בנושא יצירת תוכניות המכילות מסדי נתונים (כמו התוכנית שהוצגה בפרק זה) עיין בפרק 26, **שימוש באובייקטי גישה לנתונים (DAO)**.

18

תכנון ממשק נכון

מה בפרק?

- תכנון טפסים יעילים
- הבדלי צורת הממשק במחשבי המשתמש
- הגשמת ציפיות הלקוח מהתוכנית

בפרק זה, תלמד על חלק התוכנית שאותו כולם רואים: ממשק המשתמש. ישנם מפתחים המתייחסים לממשק המשתמש כאל חלק שולי, מאמינים כי כתיבת הקוד היא האתגר האמיתי ביצירת התוכנית ומקדישים לו את רוב המאמץ. האמת היא, שיש לשים לב לא פחות לגופנים שבהם משתמשים, צורת התצוגה ומהירות העבודה של התוכנית. המשתמשים אינם יכולים לראות את הקוד, ומצד שני ממשק המשתמש (לטוב או לרע) נמצא תמיד מול עיניהם. מערכת ההפעלה מציעה אפשרויות רבות לבניית ממשק אשר יעזור למשתמש לבצע את עבודתו בקלות רבה יותר. פרק זה מתאר שורת הנחיות ודוגמאות שיעזרו לך להפיק את המירב מאפשרויות אלו.

תכנון טפסים יעילים

טפסים מהווים למעשה אבני בניין בתוך ממשק המשתמש. למרות שדרך עיצוב הטופס בסביבת Visual Basic הינה פשוטה, לא קל לעצב ממשק המתוכנן היטב. תכנון טוב של טופס איננו מתבטא בהוספה אקראית של פקדים ושגרות אירוע. כדי ליצור טופס מעוצב היטב, עליך להבין את מהות הטופס, מה השימושים שלו, מתי ישתמשו בו והקשר שלו לשאר חלקי התוכנית. בנוסף, תוכל להכיל כמה טפסים פתוחים ביישום שלך כשכל אחד מהם מוצג בזמן הנכון. ישנם משתמשים הרגילים להשתמש ביכולת ריבוי המשימות הקיימת במערכת ההפעלה בזמן שאחרים מעדיפים להשתמש ביישום אחד כל פעם. זכור נקודה זו כאשר אתה מעצב את **ממשק המשתמש** (User Interface). עליך להשתמש בגמישות המוצעת למפתח על ידי מערכת ההפעלה כדי שכל משתמש, בכל רמת כישורים, יוכל להשתמש ביישום בצורה יעילה.

שמירה על טפסים מסודרים ויפים

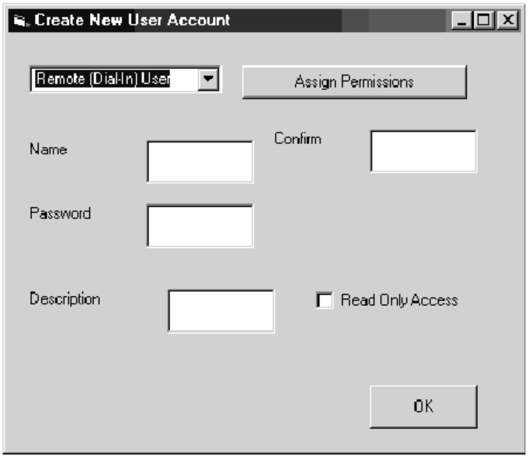
ככל שתוסיף פקדים לטופס, כך חשוב יותר לשמור שהם יהיו מסודרים. הבט בטופס המוצג בתרשים 18.1. טופס זה נראה כאילו הפקדים הוצבו בטופס בצורה לא מאורגנת. אין להם תוויות, הם לא מסודרים וגודלם שונה זה מזה. גישה עדיפה לסידור טופס מוצגת בתרשים 18.2. שים לב לכך שנוספו מסגרות, שורות ותוויות לפקדים המסודרים בטופס. שתי הדוגמאות מציגות יישומים אשר יכולים לעבוד בצורה טובה, אולם נוח יותר לעבוד עם הטופס השני בגלל הצורה החיצונית שלו.

Visual Basic מספקת כמה פקדים מצוינים שיוכלו לעזור לך ליצור טופס מאורגן. אחד מהפקדים האלה הוא פקד TabStrip, הנראה בתרשים 18.3, אשר בעזרתו ניתן לשנות תכונות פקדים כך שרק כמה מהם יוצגו בו זמנית. בצורה זו, תוכל להסתיר אפשרויות מתקדמות מעיני המשתמש הרגיל ולהשאיר אותן זמינות במידת הצורך.

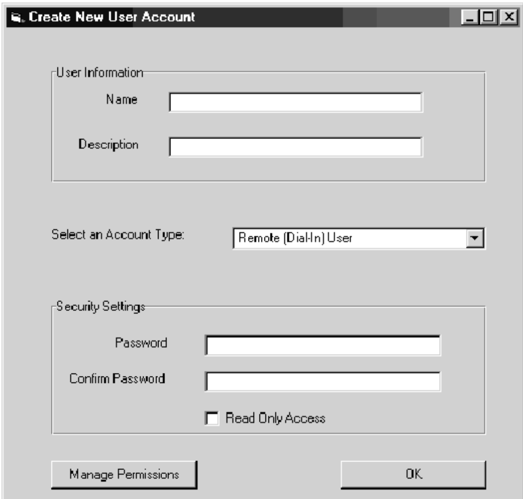
הערה:

הפקד Tabbed דומה לפקד TabStrip. ההבדל העיקרי ביניהם הוא שפקד TabStrip יכול להכיל פקדים אחרים.

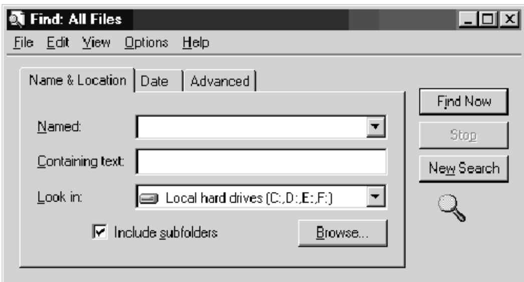
ראה: "פקד TabStrip", פרק 12.



תרשים 18.1: טופס מבולגן עלול למנוע מהמשתמש לאתר את המידע הדרוש



תרשים 18.2: מסגרות ופקדי תוויות עוזרים לסדר את הטופס בצורה טובה יותר



תרשים 18.3: השתמש בפקד TabStrip כדי לשמור על הטופס בגודל סביר

כדי להימנע מלהציף את הטופס ביותר מדי פקדים, עליך לתכנן אותו בהתאם למטרתו בתוכנית. קח לדוגמה את הטופס המוצג בתרשים 18.3. ניתן לכלול את כל הפקדים בטופס בהגדרה "אפשרויות תוכנית", אולם נוח יותר כאשר כל קטגוריה בהגדרה אפשרויות נמצא במקטע אחר של פקד TabStrip. אם תגיע למסקנה שעליך להשתמש בטופס נפרד, ודא שאתה מציב עליו פקדים בצורה הגיונית.

בנוסף, הקפד לקבוע את מאפייני הטופס המדובר כך שהוא יתנהג בהתאם למה שהוא מיועד לו בתוכנית. לדוגמה, אין צורך במתן אפשרות לשנות גודל טופס מודאלי, או להציגו בשורת המשימות של Windows.

ראה: "מבט חוזר על מאפייני הטופס", פרק 3.

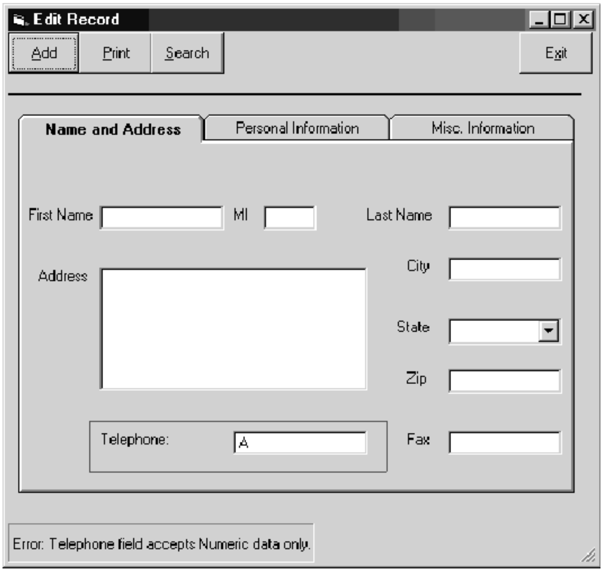
שים לב לטפסי קליטת נתונים

טפסי קליטת נתונים הינם טפסים מסוג מיוחד. הם אמורים לאפשר למשתמשים לעבוד בקצב שלהם, לא בקצב שהמפתח קבע. המפתח לתכנות נכון בטפסים אלה הוא היגיון פשוט: אם על המשתמש להזין 10000 רשומות לבסיס הנתונים, הדבר האחרון שהוא ירצה לעשות זה להשיב על שאלת כן/לא מיותרת עבור כל רשומה...

הסעיף הקודם מטפל בהפרדה והסתרה של פקדים מסוימים. אולם, טופס קליטת נתונים אמור לנצל את השימוש בשטח הטופס עד למקסימום מפני שהסתרה והצגה של פקדים גורמת להאטת התהליך. אחת מהמטרות העיקריות בתכנון טופס קליטת נתונים היא גורם המהירות. כדי לגרום לתהליך מהיר יותר של קליטת נתונים, עבוד לפי ההנחיות הבאות:

- ככלל, הוסף מקשי קיצור. **לעולם** אל **תדרוש** שימוש בעכבר (כלל זה טוב עבור כל הטפסים בתוכנית, לא רק לטפסי קליטת נתונים).
- שמור על מבנה הטופס כך שיתאים לסדר הפעולות שהמשתמש מבצע. במילים אחרות, אל תכריח את המשתמש לקפוץ קפיצות מיותרות בין חלק אחד לשני כדי להזין מידע רציף.
- אל תדרוש מהמשתמש לבצע פעולות מיותרות. במילים אחרות, אם שדות 2 עד 10 דורשים ערך מספרי רק בתנאי ששדה 1 מכיל ערך שכזה, אין צורך לאלץ את המשתמש לעבור בין כל השדות. מצד שני, אל תגרום לטפסים שאתה מעצב להתנהג בצורה עצמאית מדי. אם הטופס יתנהג בצורה שונה עבור כל שילוב אפשרי של שדות, אתה עלול להאט בכך את פעולת המשתמש.
- השתמש ברמזים שקל לראות אבל שלא יהוו מכשול כדי להגיב על פעולות המשתמש. הצורה שבה עורך הקוד של Visual Basic מתקן את האות הראשונה של קבועים ומשתנים מהווה דוגמה טובה לכך.
- הצב, במידת האפשר, פונקציות הוספה ועריכה בטופס משותף כך שהמשתמש לא יחויב ללמוד שיטות שונות כדי להגיע לאותו מידע.

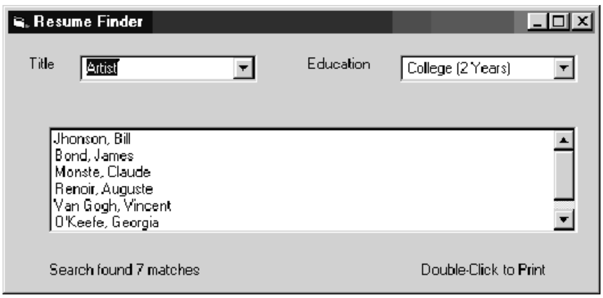
דוגמה טובה לטופס קליטת נתונים מוצגת בתרשים 18.4. שים לב לכך שכאשר מוזנים נתונים שגויים, השדה שבו הוזן המידע מודגש ומוצג הסבר בשורת המצב.



תרשים 18.4: הנה דוגמה לטופס קליטת נתונים טוב המאפשר עבודה קלה ונוחה

השתמש בפקד הנכון כדי לבצע את המשימה

Visual Basic מספקת כמה פקדים גמישים הניתנים להצבה בטופס. עם זאת, זכור שישנם פקדים העובדים טוב יותר מפקדים אחרים במצבים מסוימים. עליך להשתמש במטרה שלשמה קיים הטופס כקו מנחה בבחירת הפקדים המתאימים. לדוגמה, ניתן להשתמש בפקד ListBox ובפקד ComboBox כדי לבחור אפשרות מתוך רשימת אפשרויות. ההבדל ביניהם הוא בכך שפקד ComboBox מאפשר לחסוך בשטח הטופס על ידי הסתרת רשימת האפשרויות עד לרגע שבו המשתמש זקוק לה, כמתואר בתרשים 18.5.



תרשים 18.5: כמות השטח הפנוי בטופס עלולה להשפיע על החלטתך להציג פקדים מסוג מסוים

בתרשים 18.5, ניתן להשתמש בפקדי תיבות רשימות (ListBox) באותה מידה, אולם שימוש בתיבות מסוג Combo Box מתאים יותר מפני שבכך נחסך שטח מיותר של הטופס תוך מילוי המשימה במלואה.

ראה: "פקד ComboBox", פרק 4.

פקדים של ספקים חיצוניים

פקדים של ספקים חיצוניים שימושיים ביותר, אולם אין להשתמש בהם אלא אם הדבר נחוץ. שימוש בפקד פנימי של Visual Basic לעומת שימוש בפקד שהגיע ממקור חיצוני מציע את היתרונות הבאים:

- רוב הסיכויים הם שהפקד ייתמך על ידי גרסאות עתידיות של Visual Basic.
- הפצת הפקד למשתמשים נעשית בצורה קלה יותר.
- פקדים פנימיים רבים, כגון Windows Common Control, מספקים למשתמש ממשק שהוא כבר רגיל לעבוד איתו.

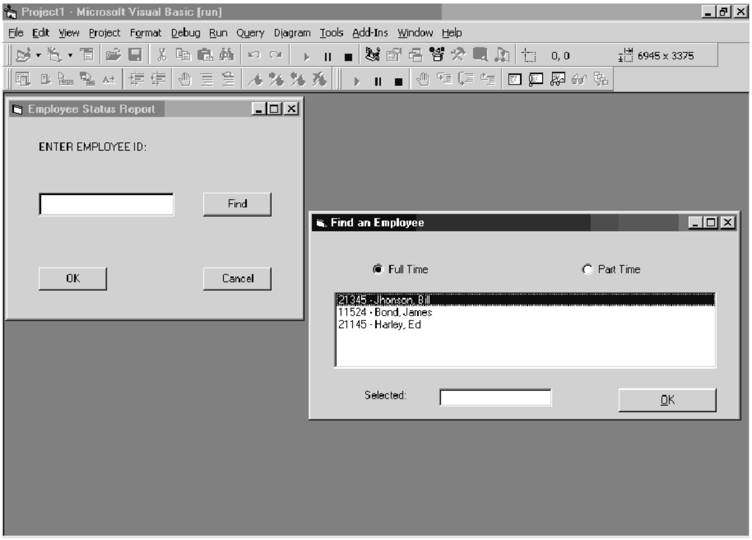
ריבוי טפסים

במידה וממשק המשתמש מכיל כמה טפסים, יש להחליט החלטה חשובה בקשר לשימוש ב**ממשק בעל טופס יחיד** (SDI – Single Document Interface) או בממשק מרובה טפסים (MDI – Multiple Documents Interface). יישומים בעלי ממשק MDI, המתוארים בפרק 17, **יישומים בעלי ממשק מרובה מסמכים**, מטפלים בנושא ריבוי הטפסים על ידי הכלת כל טפסי היישום בתוך טופס אב.

ראה: "מבוא ליישומי MDI", פרק 17.

טפסים ביישומי SDI מופיעים כחלונות עצמאיים לגמרי. בשני סוגי היישומים, MDI ו-SDI, הקשר של המשתמש עם הטפסים גורם להפעלת פונקציות רבות בתוכנית, על ידי התרחשות אירועי פקדים וטפסים. אם קיימים מספר טפסים, עליך לארגן את קוד התוכנית שידאג לכך שהמשתמש לא יוכל להפריע למהלך התקין של פעולת התוכנית. לדוגמה, יש לדאוג לכך שלא יוצג טופס המכיל מידע שלא מיועד עדיין להצגה.

כדוגמה לבעיה אפשרית בשימוש בטכניקת ריבוי טפסים, הבט בשני הטפסים המוצגים בתרשים 18.6.



תרשים 18.6: טפסים שהקוד מתייחס אליהם כטפסים עצמאיים בזמן שקיים ביניהם קשר, עלולים ליצור בעיות לא צפויות

בדוגמה המוצגת בתרשים 18.6, משתמשים בטופס frmMain כדי לאחזר מידע מתוך בסיס הנתונים. אם המשתמש אינו יודע את מספר הלקוח, ניתן ללחוץ על לחצן **Find** הנמצא ליד השדה כדי להציג את הטופס השני, **Find an Employee**. טופס זה מוצג כאשר הוא מכיל ערך ID קיים ומחזיר את הערך הרצוי לטופס הראשי. במבט ראשון, קל להבין את הקוד המבצע את ההעברה בין הטפסים.

תפקיד לחצן החיפוש בטופס הראשון הוא בסך הכל להציג את הטופס השני (טופס החיפוש):

```
Sub cmdSearch_Click()
	frmSearch.Show
End Sub
```

לאחר שבוצעה פונקציית החיפוש, דואג הטופס השני להציב את הערך הסופי בתיבת הטקסט שנמצאת בטופס הראשון. לאחר מכן הטופס מוחק את עצמו מהזיכרון:

```
Sub cmdOK_Click()
	frmMain.txtID = sSearchResult
	Unload Me
End Sub
```

למרות העובדה שהקוד המוצג כאן יעשה את העבודה נאמנה, קיימות בו מספר בעיות עיצוב בסיסיות. קודם כל, הקוד מתייחס לטופס frmSearch כאל טופס עצמאי לגמרי. המשתמש יוכל לחזור לטופס frmMain בלי לסגור את טופס frmSearch, בין אם בכוונה ובין אם לא, ולגרום בכך לטופס יתום להישאר פתוח. אם טופס frmSearch ינסה לגשת

לטופס frmMain אחרי שהטופס השני נמחק מהזיכרון, יגרום הדבר לשגיאת מערכת. הבעיה השנייה בצורת ארגון השגרות הוא בכך שהקוד בנוי בצורה קשיחה המיועדת לשני טפסים מסוימים, ולכן יהיה קשה לעשות שימוש בטופס frmSearch בחלקים אחרים של היישום. תוכל לפתור את שתי הבעיות על ידי הצגת טופס frmSearch בצורה מודאלית (Modal).

טפסים מודאליים, כפי שזכור לך, שומרים על מיקוד היישום עד לרגע שבו הם מוסתרים. במילים אחרות, לא תוכל לעבור מטופס מודאלי לטופס אחר לפני שהטופס המודאלי נסגר.

ראה: "מבט חוזר על מאפייני הטופס", פרק 3.

הצגת טופס frmSearch בצורה מודאלית מונעת מהמשתמש מלעבור בחזרה לטופס הראשי ללא סגירת הטופס הנוכחי קודם לכן. שיטה זו גם עוצרת את הקוד הפועל בטפסים אחרים בזמן שהטופס מוצג. תוכל לנצל יכולת זו לטובתך. ראה את פונקציית החיפוש המחזירה את הערך המבוקש לשיגרה שקראה לה ולא לטופס מסוים:

```
Function sGetValidValue() As String
	frmSearch.Show vbModal
	sGetValidValue = frmSearch.sSearchResult
	Unload frmSearch
End Function
```

כעת תוכל לקרוא לשיגרה המתוארת כאן מתוך טופס frmMain או מכל טופס אחר, כדי להציג את טופס frmSearch ולקבל ערך תקין לחיפוש:

```
Sub cmdSearch_Click()
	frmMain.txtID = sGetValidValue()
End Sub
```



טיפ:

אם הפרויקט שאתה כותב גדול, השתמש במודלים, תת-שגרות ואפילו ספריות DLL כדי להפריד בין ה"קרביים" של התוכנית לממשק המשתמש, ובכך תהפוך אותה לנוחה יותר לכתיבה.

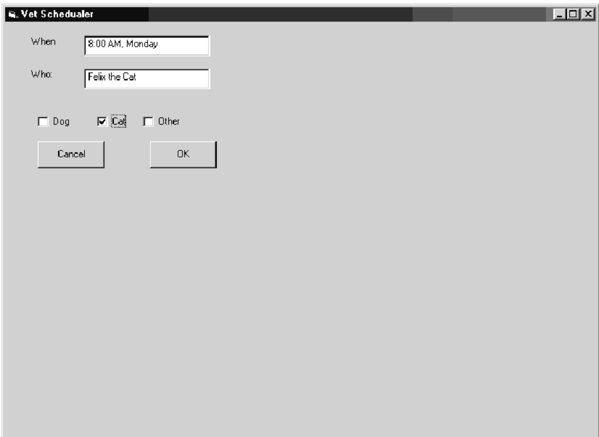
הבדלי צורת ממשק במחשבי המשתמשים

יש לנהוג משנה זהירות כאשר מסתכלים על התוכנית מנקודת המבט של ממשק המשתמש בזמן כתיבת תוכנית המיועדת לפעול על מחשב שאינו מוכר לך. תוכניות הכתובות תחת Visual Basic יכולות לפעול בסביבת Windows 95, Windows 98, או Windows NT (ועדיין לא הזכרנו את Windows CE). כל אחת ממערכות הפעלה אלו שונה מן השנייה וכל מחשב הפועל תחתן נראה אחרת.

הערה:

אם תרצה שהתוכנית תבחין בסוג מערכת ההפעלה אשר בה היא מופעלת, בדוק את פונקציית המערכת (API) GetVersionEx או את הפקד העצמאי SysInfo. ניתן למצוא הוראות שימוש בשגרות API בפרק 20, ***גישה ל-API של Windows***.

בנוסף, מערכת ההפעלה משאירה מקום רב להתאמתה בצורה אישית על ידי המשתמש. אחד ההבדלים הבולטים ביותר הוא בדרך כלל רמת ההפרדה (Resolution) של המסך. אם תגיע ללוח הבקרה של Windows, תבחין שקיימות כמה אפשרויות לשינוי התצוגה. כמפתח והבעלים המאושרים של מסך בגודל 21 אינטש, אני משתמש בהפרדה של 1280X1024 שמאפשרת לי תצוגת פרטים רבים על המסך. המשתמש הממוצע, לעומת זאת, נוהג להשתמש בהפרדה נמוכה יותר, כגון 640X480. הדרך הקלה ביותר למנוע בעיות תאימות עם לקוחות אלה היא לתכנן את כל היישום בהפרדה של 640X480. המשתמשים בעלי מסך הפועל בהפרדה זו יראו את היישום על פני כל שטח המסך בזמן שלמשתמשים שלהם הפרדת מסך גבוהה יותר יהיה שטח נוסף של שולחן העבודה הפנוי ליישומים אחרים. אולם מה יקרה אם המשתמש יבחר לשנות את גודל הטופס ביישום שלך בעזרת העכבר? במידה ולא תכתוב קוד הבנוי לטפל באירוע מסוג זה, משתמשים שלהם הפרדת מסך גבוהה יותר מהממוצע עלולים לקבל תוצאות לא נעימות לעין, כמו הטופס המוצג בתרשים 18.7.



תרשים 18.7: אם לא תכתוב קוד שיטפל בשינוי גודל הטופס על ידי המשתמש, הוא עלול לתפוס שטח מיותר רב

בנוסף, המשתמש יכול להקטין את הטופס ולגרום בכך להסתרת חלק מהפקדים. אם הטופס שלך אינו מכיל הרבה מרכיבים, תוכל להוסיף קוד לשגרת Resize כדי לשחרר אותו מתלות במימדי מסך מסוימים. ניתן לעשות זאת על ידי בדיקת מאפייני Height ו-Width של כל פקד ולהשוות אותם לגובה ורוחב הטופס. תוכל להוסיף קוד בקלות אשר ישמור על מיקומם היחסי של הפקדים כאשר המשתמש משנה את גודל הטופס. הקוד המתואר כאן משנה את גודל תיבת רשימה (ListBox) ושני לחצני פקודה כך שגודלם ישתנה באופן יחסי לגודל הטופס:

```
Private Sub Form_Resize()
    If Me.Height <= 1365 Then Exit Sub
    lstMain.Height = Me.Height – 1365
    lstMain.Width = Me.Width – 420
    cmdOK.Top = lstMain.Height + 360
    cmdOK.Left = lstMain.Width – cmdOK.Width
    cmdCancel.Top = cmdOK.Top
    cmdCancel.Left = cmdOK.Left – cmdCancel.Width – 120
End Sub
```

טיפ:

כדי ליצור טופס טוב יותר, שנה את גודל הגופן בפקדים בהתאם לגודל הטופס.

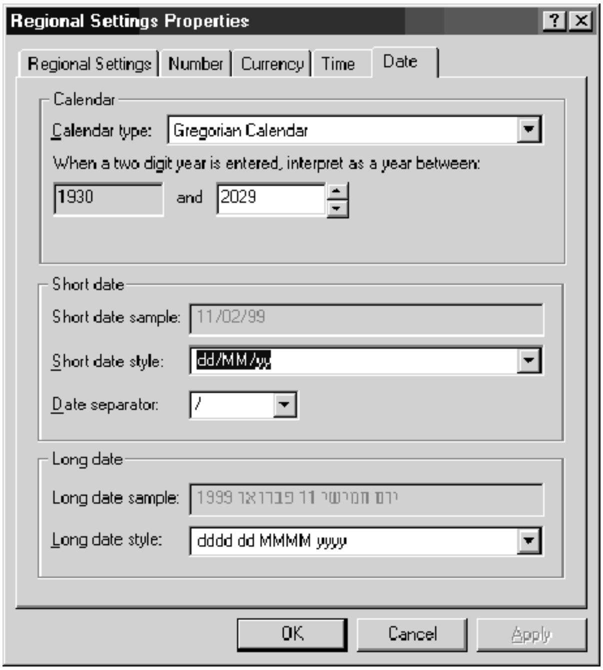
שים לב לכך ששורת הקוד הראשונה בשגרת Resize גורמת לסיום השיגרה אם גובה הטופס קטן מדי כדי לא לגרום ליצירת שגיאה במידה ומתקבל ערך שלילי.

הערה:

תוכל להשתמש בפקדים נפרדים המסופקים על ידי חברות שונות כדי לגרום לכך שהיישומים שלך לא יהיו תלויים בהפרדת המסך.

צורת התאמה אישית נוספת אשר יכולה להשפיע על התוכנית במידה רבה היא ההתאמה האזורית המוגדרת בלוח הבקרה ומוצגת בתרשים 18.8. מסך זה מאפשר למשתמש לקבוע את צורת הצגת המטבע, התאריך, ואלמנטים נוספים במערכת ההפעלה. אם אתה משתמש בהגדרות אלו בתוכניות שלך, ודא שהמשתנים, שדות בסיסי הנתונים, ופונקציות החישוב מתייחסים להן.

ראה: "תוצאות תהליך העיצוב", פרק 9.



תרשים 18.8: ודא שהתוכנית שלך ערוכה לטיפול בדרך שבה המשתמש הגדיר את ההתאמה האזורית שלו

עמידה בדרישות הלקוח

למשתמשים יש בדרך כלל ציפיות מסוימות מהדרך שבה מתנהגות תוכניות הפועלות בסביבת Windows. חלק מהציפיות הגיוניות וחלקן לא. לרוב, דרישות אלו נובעות מדוגמאות אחרות שהמשתמש נתקל בהן בעבר ביישומים אחרים. טבעי הדבר שהוא יצפה מתוכניות המבוססות על Visual Basic להתנהג באותה הצורה בדיוק.

לדוגמה, למרות שלתוכנית שלך יהיה לחצן יציאה (Exit), תוכל לסמוך על כך שחלק מהמשתמשים יעדיפו ללחוץ על לחצן הסגירה הנמצא בצד ימין למעלה של החלון. אם שכחת להוסיף קוד לשגרת Form_Unload כך שתדע לטפל במצב זה, היישום ימשיך לפעול בניגוד לרצונך. הסעיפים הבאים מתארים כמה דרכי פעולה שאתה יכול להיות בטוח שהמשתמשים מצפים להן.

תיבת הרשימה (List Box)

הרבה משתמשים מצפים מתיבות אלו להתנהג בדרך מסוימת. נניח שתיבה כזו מוצבת בתיבת דו-שיח ומכילה פריט אחד בלבד. חלק מהמשתמשים יעדיפו ללחוץ לחיצה כפולה על הפריט עצמו כדי לבחור אותו, בזמן שאחרים יסמנו אותו ויקישו על מקש Enter או שילחצו על לחצן OK. ניתן לכלול דרכים חלופיות אלו בקלות בתוכנית שלך. כל שצריך לעשות הוא לזכור להוסיף קוד לשגרת DblClick של תיבת הרשימה.

טיפ:

בדוגמה המתוארת כאן, אשר בה המשתמש בוחר פריטים מתוך תיבת הרשימה, ניתן להשתמש בלחיצה בודדת או בלחיצה כפולה כדי לבצע את המשימה. ישנם משתמשים שלא יודעים מתי הם אמורים ללחוץ לחיצה אחת ומתי נדרשת לחיצה כפולה. ניתן לזהות משתמשים אלה בקלות בזכות העובדה שהם נוטים ללחוץ לחיצה כפולה על כל דבר שקיים בטופס: קישורים, לחצני פקודה, ואפילו תפריטים. תוכל ללמד אותם לתקן את דרך פעולתם על ידי שימוש בסמן שעון החול ומתן הוראות שמגדירות איזה סוג לחיצה יש ללחוץ בכל מקרה ספציפי.

ראה: "תיבת הרשימה ", פרק 4.

מאפיין אחר של תיבת הרשימה הקיים בתוכניות מסחריות היא היכולת להשתמש במקשי קיצור במקלדת כדי "לקפוץ" לפריט מסוים ברשימה. תיבת הרשימה מקבלת כברירת מחדל רק תו אחד. תוכל להוסיף בקלות אפשרות להקשה מקוצרת של שם הפריט לתיבה כדי להגיע אליו בצורה מהירה. הדרך לעשות זאת היא לאגור את סדרת המקשים שהוקשו משגרת KeyPress בתוך מחרוזת, ולהעביר את הסמן לפריט המתאים ברשימה, כפי שמוצג בתוכנית 18.1. קוד זה גם מספק את האפשרות של הוספת פקד Timer המאפשר לבחור את הפריט המבוקש לפני סיום הקשת שמו המלא.

תוכנית 18.1: LISTBOX.VBP - הוספת האפשרות לחיפוש פריט בעזרת המקלדת.

```
Option Explicit
Dim sLstSearch As String            'The search string
Dim sCurrentLetter As String  'Letter being loaded into the list
Dim sWaitForLetter As String  'Letter not yet loaded but selected
Dim dbBiblio As Database            'The sample VB "biblio" database
Private Sub Form_Load()
        sWaitForLetter = ""
        CurrentLetter = ""
        lblLoaded.Caption = "Loading the list… "
        Set dbBiblio = OpenDatabase("d:\vb6\biblio.mdb")
        DoEvents
        tmrLoad.Enabled = True
End Sub
```

```
Function SearchListBox (lstX As Control, KeyAscii As Integer)
        Dim nSearchPos As Integer
        Dim nSearchLen As Integer
        Dim nResult As Integer

        nSearchPos = lstX.ListIndex
        nSearchLen = Len(sLstSeach)
        sWaitForLetter = ""

        'CHECK FOR BACKSPACE KEY
        If KeyAscii = vbKeyBack Then
            nSearchPos = 0
            If nSearchLen > 0 The sLstSearch = Left$(sLstSearch, nSearchLen – 1)
        End If

        'CAPITALIZE THE NEW CHARACTER
        KeyAscii = Asc(Ucase$(Chr$(KeyAscii)))

        'IF YOU PRESS 'X' AND DATA IS ONLY LOADED TO 'C' THEN WAIT….
        If nSearchLen = 0 And KeyAscii > Asc(sCurrentLetter) Then
                lstX.ListIndex = lstX.ListCount – 1
                sWaitForLetter = Chr$(KeyAscii)
                DoEvents
                Exit Function
        End If

        'ADD NEW LETTER TO SEARCH STRING
        If KeyAscii >= Asc("A") And KeyAscii <= Asc("Z") Then
                sLstSearch = sLstSearch & Chr$(KeyAscii)
        End If

        'DISPLAY SEARCH STRING
        lblSrchtext.Caption = sLstSearch

        'RE-CALCULATE LENGTH
        nSearchLen = Len(sLstSearch)

        'SIMPLE SEARCH – COMPARES EACH ITEM TO SEARCH STRING
        Do While nSearchPos < lstX.ListCount
                nResult = StrComp(sLstSearch, Ucase$(Left$(lstX.List(nSearchPos),_
                        \nSearchLen)))
                If nResult <= 0 Then
                        lstX.ListIndex = nSearchPos
                        SearchListBox = 0
                        Exit Function
```

```
            End If
            nSearchPos = nSearchPos + 1
        Loop
        'IF ITEM WAS NOT FOUND THEN MOVE TO END OF LIST
        lstX.ListIndex = lstX.ListCount – 1
        SearchListBox = 0

End Function

Private Sub lstSearch_KeyPress(KeyAscii As Integer)

        If KeyAscii = 13 Then Exit Sub
        KeyAscii = SearchListBox(Me.ActiveControl, KeyAscii)

End Sub

Private Sub tmrLoad_Timer
        Dim sSQL As String
        Dim nTemp As Integer
        Dim rs As Recordset

        'MOVE TO THE NEXT LETTER IN THE ALPHABET
        If CurrentLetter = "" Then CurrentLetter = "A" Else CurrentLetter = _
            Chr(Asc(CurrentLetter) + 1)

        'IF 'Z' ITEMS HAVE BEEN LOADED THEN STOP THE TIMER
        If CurrentLetter > "Z" Then
            tmrLoad.Enabled = False
            lblLoaded = CStr(lstSearch.Liscou) & " records "
            dbBiblio.Close
            Screen.MousePointer = vbDefault
            lstSearch.SetFocus
            Exit Sub
        End If

        'QUERY THE DATABASE AND ADD RECORDS TO THE LIST BOX
        sSQL = "SELECT Name From Publishers where Name Like '" & CurrentLetter & _
                "*' order by Name"
        Set rs = dbBiblio.OpenRecordset(sSQL)
        Do While Not rs.EOF
            lstSearch.AddItem CStr(rs.Fields("Name"))
            rs.MoveNext
        Loop
        rs.Close
        lblLoaded = "Loading " & lstSearch.ListCount & " records ..."
```

```
        lstSearch.SetFocus
        DoEvents

        'DISPLAY HOURGLASS IF WAITING ON DATA
        If sWaitForLetter = "" Then
                Screen.MousePointer = vbDefault
                Exit Sub
        Else
                Screen.MousePointer = vbHourglass
                nTemp = SearchListBox(lstSearch, Asc(Trim$(sWaitForLetter)))
        End If
End Sub
```

תפריטים יעילים

חלק חשוב נוסף בעיצוב הטופס הוא ליצור תפריטים יעילים ומובנים. הנה כמה הנחיות חשובות לתהליך העיצוב:

- השתמש במערכת הנפוצה של תפריטי יישומים במערכת הפעלה Windows: **File**, **Edit**, **View** וכדומה.
- קבץ אפשרויות בתפריטים בצורה הגיונית ומובנת.
- השתמש במפרידים בתפריט נפתח (Drop-down menu) כדי לקבץ פריטים השייכים לקבוצה מסוימת.
- הימנע מהצגת אפשרויות מיותרות.
- הימנע משימוש באפשרויות המוצגות על שורת התפריט שאינן מכילות תפריט.
- זכור להשתמש בשלוש נקודות (...) כדי לסמן פריטים בתפריט אשר גורמים להופעת תיבת דו-שיח.
- השתמש בקיצורי דרך ובמקשי קיצור סטנדרטיים ככל שניתן.
- הצב את האפשרויות הנפוצות בתפריט על גבי סרגל כלים.

צעדים אלה הקשורים לשימוש בעורך התפריטים וביצירת תפריט נדונים בפרק 6, **שליטה נוספת למשתמש: תפריטים וסרגלי כלים.**

ראה: "יצירת תפריטים", פרק 6.

טיפול במספר מופעים פעילים של היישום

תוכניות מבוססות Windows רבות מופעלות כאשר המשתמש לוחץ על קיצור הדרך שלהם לחיצה כפולה או בוחר אותן מתפריט. כאשר יש למשתמש עשר תוכניות פתוחות, הוא יכול לעבור ביניהן על ידי שימוש בשורת המשימות (Task Bar) או על ידי שימוש במקשי הקיצור Alt+Tab. למרות זאת, הרבה משתמשים מעדיפים ללחוץ לחיצה כפולה על סמל קיצור הדרך פעם נוספת אפילו אם התוכנית נטענה כבר לזיכרון. כמה מקיצורי הדרך, כגון סמל **המחשב שלי** הקיים במערכת Windows 95, גורמים להצגת החלון הקיים כבר בזיכרון, סמלים וקיצורי דרך אחרים גורמים ליצירת מופע חדש של היישום בזיכרון. לדוגמה, תוכל לפתוח כמה מופעים של מעבד התמלילים Word על ידי בחירת הסמל המתאים בתפריט **התחל**, אולם לחיצה כפולה על סמל השייך למסמך Word מפעילה את העותק הפעיל בזיכרון ומשתמשת בו. כדי להמשיך ולסבך את העניין, סרגל הכלים של Microsoft Office בודק אם יש עותק פעיל של Word ומציג אותו במקום ליצור מופע חדש.

אם מספק קיצור דרך ליישום על שולחן העבודה או על סרגל הכלים, המשתמש יצפה שהוא יפעיל את העותק הפעיל בזיכרון במקום ליצור מופע חדש. דבר זה נכון במיוחד אם המשתמש מילא טופס מסוים ומזער אותו לאחר מכן בשורת המשימות. במצב זה הוא עלול ללחוץ לחיצה כפולה על קיצור הדרך ולצפות לכך שהטופס הקודם יופיע, אולם אם לא דאגת לכתוב קוד המטפל במצב מעין זה, יופיע מופע חדש של היישום והמשתמש ייראה טופס ריק.

אפשור יצירת מספר מופעים של היישום בזיכרון יכולה להיות רצויה עבור חלק מהיישומים, אולם אם לא תתחשב באפשרות זו בזמן כתיבת התוכנית אתה עלול לגרום לשגיאות. לדוגמה, מספר מופעים של התוכנית עלולים לנסות ולכתוב מידע לאותו בסיס הנתונים באותו זמן. למזלנו הטוב, מניעת האפשרות ליצור מופעים נוספים של התוכנה בזיכרון קלה לביצוע. שורות הקוד הבאות בשגרת Form_Load או Sub Main יעשו את העבודה:

```
If App.PrevInstance = True then
      MsgBox "Application already running!"
      End
End If
```

שורות הקוד המתוארות כאן בודקות את מאפיין PrevInstance של אובייקט היישום ומפסיקות את פעולת התוכנית אם קיים מופע נוסף. למרות שהקוד מונע טעויות והתנגשויות תוכנה, הוא אינו עוזר למשתמש מפני שהוא צריך עדיין למצוא את המופע הפעיל בצורה ידנית. תוכל לגרום לתוכנית שלך להציג את המופע הקודם לפני סגירת המופע החדש בעזרת קריאה לכמה שגרות API. תוכנית 18.2 מכילה שיגרה פשוטה המציגה את המופע הקודם של התוכנה.

תוכנית 18.2: PREVINST.VBP - טיפול במופעים נוספים של התוכנה.

```
Option Explicit
Private Declare Function FindWindow Lib "user32" Alias "FindWindowA"(ByVal _
        lpClassName As String, ByVal lpWindowName As String) As Long
Private Declare Function ShowWindow Lib "user32" (ByVal hWnd As Long, ByVal _
        nCmdShow As Long) As Long
Private Declare Function SetForegroundWindow Lib "user32" (ByVal hWnd As Long)  _
        As Long
Private Const SW_RESTORE = 9

Private Sub Form_Load()
        Dim sTitle As String
        Dim hwnd As Long
        Dim lRetVal As Long

        If App.PrevInstance Then
                sTitle = Me.Caption

                App.Title = "newcopy"
                Me.Caption = "newcopy"

                hwnd = FindWindow(vbNullString, sTitle)
                If hwnd <> 0 Then
                        lRetVal = ShowWindow(hwnd, SW_RESTORE)
                        lRetVal = SetForegroundWindow(hwnd)
                End If
                End
        End If

End Sub
```

בקוד המתואר, שונו שם היישום ומאפיין Caption שלו כך ששגרת FindWindow,API , לא תוכל למצוא אותו. אחר נקראו שלוש שגרות API נוספות כדי למצוא את החלון השייך ליישום הקודם, לשחזר אותו ולהביא אותו לקידמה. ניתן לקבל מידע נוסף על שגרות API של Windows בפרק 20, **גישה ל-API של Windows**.

הערה:
השיטה המתוארת כאן לטיפול בכמה מופעים של אותו יישום מתייחסים לפרויקט EXE רגיל. הטיפול בספריות ActiveX DLL מתואר בפרק 16, **מחלקות: שימוש חוזר ברכיבים**.

ראה: "קביעת מאפיין Instancing", פרק 16.

מהירות פעולה מדומה

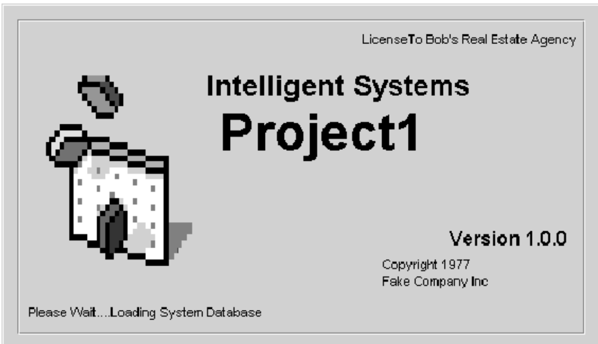
מהירות מדומה משמעה מציאות. אני מתייחס כאן לדרך שבה **נקודת המבט** של המשתמש על התוכנית יכולה להשפיע על חיבתו לתוכנית או סלידתו ממנה. מהירות פעולת היישום הינה דוגמה ראשית לכך. גם אם כתבת את הקוד המהיר ביותר שניתן לכתוב תחת Visual Basic, אין הדבר משנה אם המשתמש חושב שהתוכנית איטית. מפתחי Visual Basic נוטים להתגונן כאשר המשתמשים מתלוננים על בעיות מהירות מפני ״שהמשתמשים לא יודעים מה התוכנית עושה באמת״. תוכל להשתמש בכמה טריקים כדי לגרום לכך שהתוכנית תיראה מהירה יותר למשתמש.

המפתח למהירות מדומה של פעולת התוכנה הוא בכך שמשהו חייב לקרות כאשר המשתמש בוחר סמל. הוא יהיה נכון יותר לחכות אם הוא סבור שהמחשב עובד בצורה המהירה ביותר. אתחול Windows הוא דוגמה טובה לכך. העניין נמשך זמן רב בדרך כלל. מצד שני, כל הצלילים, שינויי התצוגה והרעשים המגיעים מהכונן מעסיקים את המשתמש מספיק זמן כדי שזמן האתחול ייראה לו סביר. הטכניקות המתוארות בסעיפים הבאים יציעו לך דרכים להפוך את התוכניות שלך ל״מהירות״ יותר.

זמן טעינת התוכנית

כאשר התוכנית מתחילה לפעול, תהיינה לה בוודאי כמה פעולות אתחול חיוניות כגון פתיחת בסיס נתונים הנמצא ברשת. שגרת Sub Main היא מקום מצוין להצבת כל קוד האתחול הדרוש בזמן טעינת התוכנית. אם לתוכנית יש מספר טפסים בלבד (שניים עד חמישה), תוכל לטעון את כולם משגרת Sub Main כך שהם יופיעו במהירות כאשר יהיה צורך להציגם. תהליך טעינת כל הטפסים יגרום להאטת ביצועי התוכנית בזמן טעינתה לזיכרון, אך מהירותה לאחר מכן תהיה גבוהה יותר. שגרת Load יוצרת את הטפסים בזיכרון, אולם הם נשארים מוסתרים מהמשתמש עד להפעלת שיטת Show.

מצד שני, טכניקה זו עלולה לגרום לזמן טעינה ארוך מאוד של התוכנית, כך שהצגת **מסך פתיחה** (Splash screen) בזמן זה יכולה להוות רעיון טוב מאוד. מסך מעין זה (ראה תרשים 18.9) מציג מידע אודות התוכנית והאדם (או קבוצת האנשים) שכתב אותה וכן מראה למשתמש שמתרחשת פעולה מסוימת.



תרשים 18.9: מסך הפתיחה מספק למשתמש משהו להתבונן בו בזמן טעינת התוכנית

טיפ:

אם אתה מעדכן שם תווית במסך פתיחה או ב- Status Screen, **ודא שתתבצע** קריאה לשגרת DoEvents או לשגרת Refresh של התווית כדי שתהליך אתחול התוכנית לא ימנע מהתווית מלהתעדכן.

שתף את המשתמש בהתקדמות התוכנית

במצב אשר בו היישום נראה כאילו הוא מבצע פעולה כלשהי, המשתמשים נוטים להיות סלחניים יותר אם עליהם לחכות זמן ממושך. דרך אחת לשתף אותם בתהליך היא להציג פקד מד התקדמות (ProgressBar) בטופס. אם התוכנית מעדכנת רשומות בבסיס נתונים, תוכל להשתמש בסרגל כזה כדי להציג את מספר הרשומות אשר עודכנו. כדי לעשות זאת, עליך להוסיף שורת קוד או שתיים אשר תדאגנה לעדכון מד ההתקדמות בכל פעם שהתוכנית תעבור לרשומה הבאה.

ראה: "הוספת מד התקדמות לתוכנית ", פרק 12.

לעיתים, האפשרות של שימוש במד התקדמות אינה סבירה. לדוגמה, ביצוע פקודת FileCopy של Visual Basic עלול לקחת זמן מה בהתחשבות בגודל הקובץ המועתק, אולם FileCopy היא פקודה עצמאית ולפיכך אין לך שום מקום אשר בו תוכל להכניס את הקוד הדואג לעדכון מד ההתקדמות. במקרה מעין זה, תוכל להשתמש בסרטון וידאו כתחליף נוח. לפני שאתה מפעיל את העתקת הקובץ, הצג סרטון וידאו בעזרת פקד האנימציה. Windows 95 משתמשת בטריק זה בזמן העתקת קבצים וריקון סל המיחזור. המשתמשים רואים סרטון וידאו וסבורים שהסרטון מקושר לתהליך העתקת הקובץ בזמן שלמעשה זהו תהליך המתבצע בנפרד מתהליך העתקת הקובץ.

ראה: "שימוש בפקד אנימציה", פרק 12.

מכאן...

למרות שאף צעד שתנקוט אינו יכול להבטיח ממשק משתמש מושלם, ממשק גרוע יכול להבטיח בוודאות שאף אחד לא ירצה להשתמש בתוכנית. למרות זאת, הקריטריון להגדרת ממשק משתמש כ״טוב״ ימשיך להשתנות תמידית עם עדכון דרכי העבודה במחשב. קח לדוגמה את תהליך כיוונון השעון במכשיר וידאו. מכשירי וידאו ישנים היו מתוכנתים בעזרת לחצנים ומפסקים, אולם שיטה זו הוחלפה במהרה על ידי תצוגת מסך. כעת אות רדיו אלחוטי הופך את התהליך כולו לאוטומטי בדגמים מסוימים. כמו בדוגמת מכשיר הוידאו, ממשק המשתמש שלך ישתנה עם הזמן ככל שהתקן בתעשיית התוכנה קובע סטנדרטים חדשים וככל שתלמד איך לעמוד בדרישות בצורה טובה יותר.

לקבלת מידע נוסף בנושאים הנדונים בפרק זה, קרא את הפרקים הבאים:

- אם תרצה ללמוד את ההבדלים בין הפקדים הזמינים לך, ראה פרק 4 **שימוש בפקדי ברירת המחדל של Visual Basic**.
- לקבלת מידע נוסף על יצירת תפריטים ואלמנטים אחרים בממשק המשתמש, ראה פרק 6 **מתן שליטה נוספת למשתמש: תפריטים וסרגלי כלים**.
- כדי לקבל מידע נוסף על יצירת רכיבי ActiveX בעזרתמודולי מחלקה, ראה פרק 16 **מחלקות: שימוש חוזר ברכיבים**.
- כדי להבין איך לפתח יישום MDI, ראה פרק 17 **יישומים בעלי ממשק מרובה מסמכים**.
- כדי ללמוד את נושא עיצוב ממשק המשתמש בצורה מפורטת יותר, ראה פרק 19 **שימוש במרכיבי התכנון הוויזואלי**.

19

שימוש במרכיבי התכנון הוויזואלי

מה בפרק?

- שימוש בגרפיקה
- עבודה עם טקסט וגופנים

בפרק 18, **תכנון ממשק נכון**, למדת כיצד לעצב את המרכיב הוויזואלי של התוכנית כדי לספק ממשק שימושי ואסתטי. פרק זה ממשיך לדון באותו נושא, ומציג בפניך חלק ממרכיבי העיצוב בהם תוכל להשתמש, כדי להעניק לתוכניתך חזות מקצועית.

שימוש בגרפיקה

ממשק משתמש טיפוסי של יישום מורכב מתפריטים, תוויות, תיבות טקסט, לחצני פקודה ואולי גם מספר פקדים מיוחדים לנתונים הייחודיים לאותה תוכנית. אולם ללא שימוש בגרפיקה, ממשק המשתמש גם שהוא שימושי, יכול להיות משעמם למדי ולא אינטואיטיבי. גרפיקה יכולה לשפר את ממשק המשתמש בדרכים הבאות:

- הדגשת מידע מסוים על המסך.
- יצירת מבט שונה על הנתונים, לדוגמה שימוש בגרף.
- יצירת קשר אינטואיטיבי יותר עם פונקציות התוכנה.

נושא העיצוב והשימוש בגרפיקה רחב ומורכב. לכן ברור כי פרק זה אינו יכול לכסות נושא זה במלואו. עם זאת, הוא יספק לך מספיק מידע שיאפשר ליצור ממשק משתמש מושך יותר מבחינה ויזואלית.

ב- Visual Basic באפשרותך להשתמש בגרפיקה בשתי דרכים עיקריות: **פקדים ושיטות**. לדוגמה אתה יכול לצייר קו על טופס על ידי הצבת פקד Line על הטופס, או באמצעות שימוש בשיטת Line.

הערה:

ממשק פיתוח היישום (**API**) של Windows מכיל גם הוא פונקציות רבות הקשורות לגרפיקה. ראה פרק 20 **גישה ל-API של Windows**, לפרטים לגבי השימוש בפונקציות אלו.

פקדי גרפיקה עובדים בדיוק כמו כל "פקד מותאם" אחר, יש להם שגרות ומאפיינים, ואפשר לציירם במהלך העיצוב בעזרת העכבר. שיטות גרפיקה הן פונקציות אשר אפשר לקרוא להם מתוך Visual Basic במהלך פעולת היישום. למרות שאתה יכול לחשוב על פקדי גרפיקה כעל אובייקטים **המונחים** על גבי הטופס, שיטות גרפיקה לעומתם מציירות על הטופס עצמו.

פקדי גרפיקה

ארגז הכלים של Visual Basic כולל מספר פקדי גרפיקה. חלק מפקדים אלה, כגון פקד Line (קו) הם פקדים פשוטים המיועדים לפעול כהרחבות ויזואליות בפני עצמם. אחרים, כגון פקד PictureBox (תיבת תמונה) כוללים מיגוון מאפיינים ושיטות כדי לאפשר תפקוד נרחב יותר. כל אחד מפקדי הגרפיקה של Visual Basic יתואר בסעיפים הבאים.

פקד Line ופקד Shape

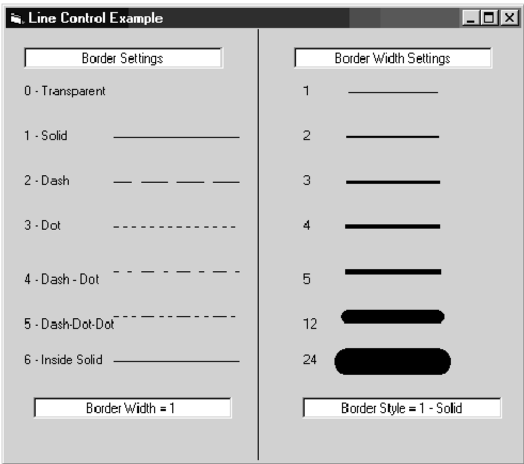
פקדי Line (קו) ו-Shape (צורה) מספקים את האמצעי הפשוט ביותר להוספת אלמנטים גרפיים לטופס. אתה יכול לצייר את הפקדים על גבי הטופס בזמן העיצוב, ולהניח אותם במקום בו אתה זקוק להם. במהלך פעולת התוכנית, ניתן להסתיר או להזיז פקדים אלה. בעזרת הגדרת ערכי המאפיינים הנכונים בקוד שלך, הינך יכול לשנות את צבעם.

כפי שניתן לנחש על פי שמו, פקד Line יוצר קו על גבי הטופס. אתה יכול לשלוט בעובי הקו, סגנונו, צבעו ומיקום נקודות הקצה שלו באמצעות מאפייני הפקד. לדוגמה, התחל פרויקט Standard EXE ומקם פקד Line על הטופס. לאחר מכן, הכנס את הקוד הבא בשגרת אירוע Resize (שנה גודל) של הטופס:

```
Private Sub Form_Resize()
        'A FORM'S (0,0) COORDINATES
        'ARE IN ITS UPPER-LEFT CORNER

        Line1.Y1 = Form1.ScaelHeight / 2
        Line1.X1 = 0
        Line1.X2 = Form1.ScaleWidth
        Line1.Y2 = Line1.Y1
End Sub
```

הקוד בשגרת אירוע Resize מכוון את גודלו ומיקומו של הקו כך שהוא נשאר קו אופקי במרכז הטופס. כברירת מחדל, יוצר פקד Line קו רציף, אך אתה יכול ליצור קו מקווקו או קו בסגנונות אחרים על ידי הגדרת מאפיין BorderStyle (סגנון גבול). תרשים 19.1 מראה מספר פקדי Line מצוירים על טופס, תוך שימוש בסגנונות שונים ואפשרויות שונות במאפיין BorderStyle של פקד Line.

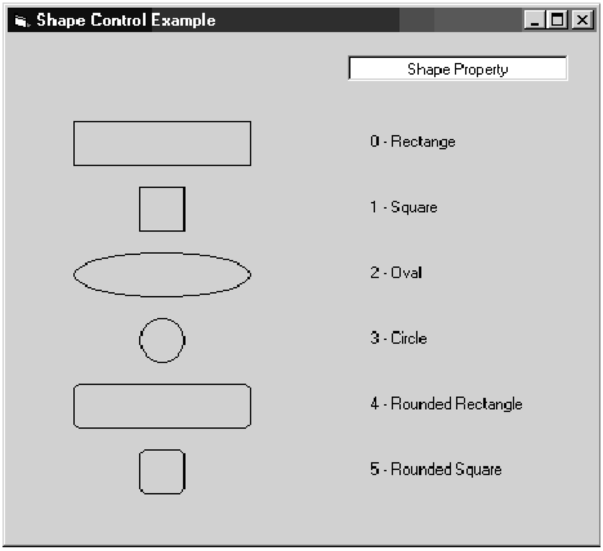


תרשים 19.1: פקד Line מאפשר לך למקם קווים בסגנונות שונים על טופס

הערה:

למאפיין BorderStyle אין כל השפעה כאשר ערך המאפיין BorderWidth גדול מהערך 1.

פקד Shape מספק אמצעי פשוט נוסף למיקום אלמנטים גרפיים על גבי טופס. ניתן להשתמש בפקד Shape כדי ליצור כל אחת מששת הצורות המוצגות בתרשים 19.2. בדומה לשימוש בפקד Line, אתה יכול להיעזר במאפיין BorderStyle לשינוי סגנון הקו המשמש לציור הצורה.



תרשים 19.2: אתה יכול לשנות את מאפיין Shape של פקד Shape כדי ליצור כל אחת מהצורות המוצגות

הערה:

למרות שפקד Shape הממוקם מאחורי קבוצת פקדים עשוי להיראות כאילו הוא "אורז" פקדים אחרים בתוכו, הוא אינו יכול לפעול כ**מכולה** לפקדים אלה.

למרות שניתן להגדיר את פקד Shape כך שיופיע במספר צורות שונות, הוא אינו משתמש בנקודות ציון מתמטיות כפי שעושה פקד Line. במקום זאת אתה מגדיר את מאפייני הגובה (Height) והרוחב (Width) כדי לקבוע את גודל הצורה.

תמונות ודמויות

דרך נוספת להוספת גרפיקה לממשק המשתמש היא למקם תמונות על הטופס. ניתן לטעון תמונות אלו לתוך פקדי Image (תמונה) ו-PictureBox או לטופס עצמו. מספר מקורות אפשריים לתמונות הינם ציורים אשר יצרת באמצעות תוכנת ציור או תמונות מסרט צילום 35mm אשר הסבת לקבצי מחשב על ידי שימוש בסורק. סוגי הקבצים שבאפשרותך להשתמש בהם מפורטים בטבלה 19.1.

טבלה 19.1: סוגי קבצים גרפיים המתאימים לעבודה ב- Visual Basic

סיומת הקובץ	**סוג הקובץ**
.BMP	(Bitmap) קובץ מפת סיביות של Windows
.DIB	(Device Independent Bitmap File) קובץ מפת סיביות ללא תלות בהתקן
.CUR,.ICO	(Icon) סמל
.WMF	(Windows Metafile) קובץ גרפי כללי של Windows
.EMF	(Enhanced Windows Metafile) קובץ גרפי כללי מורחב של Windows
.GIF	תבנית לחילופי נתונים גרפיים (Graphic Interchange Format) סוג קובץ גרפי אשר פותח לראשונה על ידי CompuServe ואשר משמש כיום בדפי Web
.JPG	קבצי JPEG, הנקראים על שם Joint Photographic Experts Group קבוצת מומחי הצילום המאוחדת, דומים לקבצי GIF אך משתמשים בדחיסה להקטנת גודל. משמשים בצורה נרחבת בדפי Web

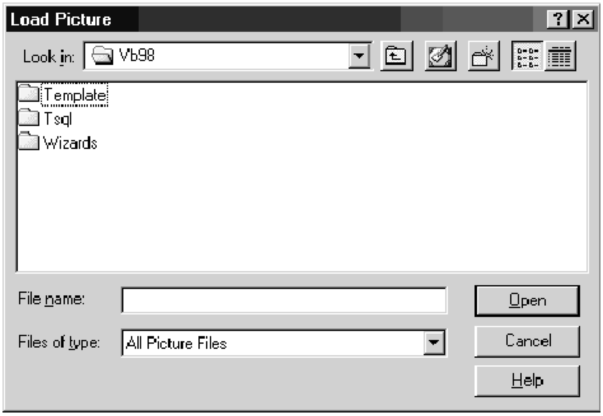
מאפיין Picture

לפקדי PictureBox, Image, ולטופס יש מאפיין בשם Picture אשר ניתן להגדירו בזמן העיצוב או בזמן פעולת היישום. טעינת קובץ תמונה בזמן העיצוב היא פעולה פשוטה. נסה דוגמה פשוטה של מיקום תמונה על טופס, כפי שמוצג להלן:

1. התחל פרויקט חדש, ופתח את חלון **Properties** (מאפיינים) של הטופס.
2. לחץ על הלחצן **בונה** (שעליו 3 נקודות) בקצה הימני של המאפיין Picture. כדי להעלות את תיבת הדו-שיח **Load Picture** (טען תמונה), המוצגת בתרשים 19.3.
3. בחר קובץ מתוך תיבת הדו-שיח **Load Picture**. התמונה הנבחרת תופיע על גבי הטופס.

הערה:

קבצים אשר נטענים בשלב העיצוב מאוחסנים ביחד עם הטופס שלך, דבר העשוי להגדיל את נפח התוכנית ולהאריך את משך זמן טעינת הטופס.



תרשים 19.3: כדי למקם תמונה על טופס, בפקדי PictureBox **או** Image בשלב העיצוב, השתמש בתיבת הדו-שיח Load Picture. ניתן גם לטעון תמונות במהלך פעולת התוכנית

בזמן שהתוכנית פועלת ניתן להעביר תמונות מפקד אחד למשנהו על ידי הצבת מאפיין Picture, או על ידי טעינת התמונה מקובץ התואם לאחד הסוגים המפורטים בטבלה 19.1. כדי להגדיר את מאפיין Picture כך שיטען קובץ גרפי, השתמש בשיטת LoadPicture. דוגמת הקוד הבאה מדגימה כיצד לטעון תמונות לטופס, לפקד Image או לפקד PictureBox במהלך פעולת התוכנית:

```
'Example of Loading pictures
Form1.Picture = LoadPicture("C:\MYPIC.BMP")
Image1.Picture = LoadPicture("D:\PAMELA.JPG")
Picture1.Picture = Image1.Picture
Picture1.Picture = LoadPicture("")
LoadPicture ("")
Set form1.Picture = "C:\MYPIC.BMP"
```

שים לב כי העברת מחרוזת ריקה לפקד LoadPicture מנקה את התמונה הנוכחית. ל- Visual Basic יש גם פקודת SavePicture (שמור תמונה) המשמשת לשמירת תמונה מתוך מאפיין Picture לקובץ בדיסק. עם זאת קיימות מספר הגבלות על סוגי הקבצים אשר ניתן לשמור. למרות שפונקציית LoadPicture עובדת עם קבצי GIF ו-JPG, פונקציית SavePicture אינה תומכת בשמירה לקבצים מסוגים אלה. תמונות הנשמרות מתוך פקד Image נשמרות תמיד כקובץ BMP. לרשימת ההגבלות המלאה על סוגי הקבצים, חפש את הפונקציה SavePicture באינדקס העזרה (Help/Index).

טעינת תמונות לטופס

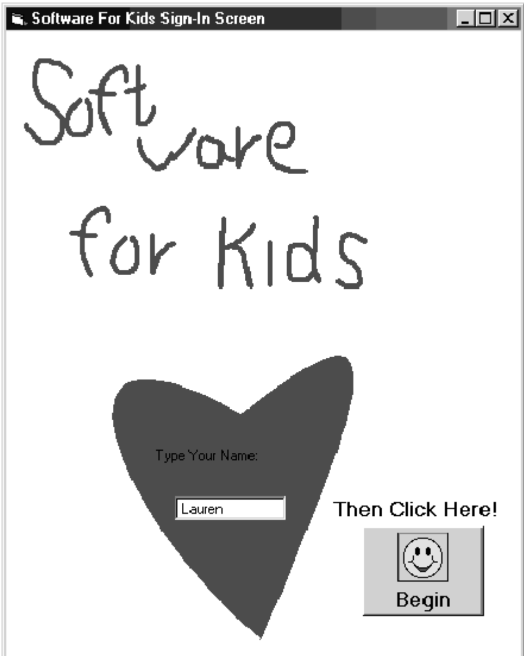
תמונה הנטענת לתוך טופס, מתנהגת בצורה סטטית יחסית, בדומה לטפט הרקע של Windows. התמונה מופיעה בפינה השמאלית עליונה של הטופס, ואינה מוגדלת בצורה כלשהי כדי להתאים לגודל הטופס. אם התמונה קטנה מגודל הטופס, נשאר מקום ריק מתחתיה ולימינה. אם התמונה גדולה מהטופס, יוצג רק אותו חלק ממנה הניתן להצגה לפי גודל הטופס, למרות שכל התמונה נטענה. כל שינוי בגודל הטופס ישפיע על החלק היחסי של התמונה אשר יוצג בתוכו.

למעט פקדי Label (תווית) ו-Shape, התמונה אינה נראית מבעד לרקע הפקדים הנמצאים על הטופס. פקדי Label ו-Shape מאפשרים לתמונה להיראות דרכם, אם מאפיין BackStyle (סגנון רקע) של הפקד מוגדר Transparent (שקוף). תרשים 19.4 מראה פקדים הממוקמים על טופס המכיל תמונה. שים לב כיצד תמונת הרקע נראית מבעד לפקד Label שמעל תיבת הטקסט.

טיפ:

לא ניתן לשנות גודל תמונה המונחת ישירות על טופס. אם ברצונך לשנות את גודלה, עליך לשנות את קובץ הגרפיקה המקורי על ידי שימוש בתוכנה גרפית

היתרון העיקרי להצבת התמונה ישירות על טופס הוא ששיטה זו משתמשת בפחות משאבי מערכת מאשר מיקום התמונה בפקד Picture או Image. יתרון נוסף להצבתה על הטופס, הוא שאפשר להשתמש בשיטות ציור כדי לכתוב הערות על התמונה.



תרשים 19.4: מספר פקדים הממוקמים על טופס, אינם מאפשרים לתמונה ברקע להיראות דרכם

לדוגמה, אתה יכול להציג תמונה של מוצר מסוים ואז על ידי שימוש בשיטת Print לכתוב את המחיר או כל נתון רלוונטי אחר על התמונה. יכולת זו קיימת גם באמצעות שימוש בפקד PictureBox אך לא עם פקד Image.

טיפ:

מיקום תמונה ישירות על הטופס היא דרך מצוינת ליצור טקסטורה או רקע לשאר האובייקטים המוצגים על הטופס.

עם זאת, הצבת התמונה ישירות על הטופס גוררת עמה מספר חסרונות, כגון:

- לא ניתן להסתיר את התמונה, אפשר רק לטעון או להסיר אותה לחלוטין.
- אין אפשרות לשלוט על מיקומה על הטופס.
- אי אפשר למקם יותר מתמונה אחת באותו הזמן על הטופס.
- לא ניתן לשנות את גודלה, היא ממוקמת על הטופס בגודלה המקורי כפי שנשמר בקובץ הגרפי בדיסק.
- ניתן להתגבר על מגרעות אלו על ידי שימוש בפקדי Picture ו-Image.

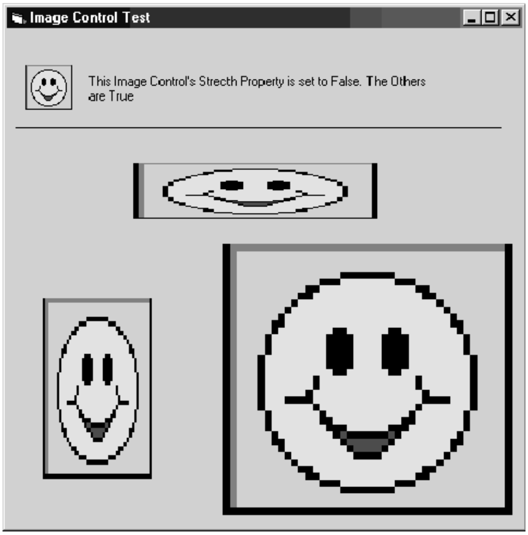
פקד Image

שימוש בפקד Image לצורך הצגת תמונות מספק מסגרת לתמונה, המאפשרת למקם אותה בכל מקום על הטופס. יתרון נוסף הוא שפקד Image מסוגל לשנות את גודל התמונות בנוסף להצגתן. מאפיין Stretch (מְתַח) קובע האם גודל הפקד ישונה כך שיתאים לגודל התמונה, או שגודל התמונה ישונה כך שיתאים לגודל הפקד. אם מאפיין Stretch מוגדר False (ברירת המחדל), גודל הפקד מוגדר מחדש באופן אוטומטי כך שיתאים לגודל התמונה שבחרת. אם מאפיין Stretch מוגדר True, גודל התמונה מוגדר מחדש כך שתתאים לגבולות הפקד כפי שהוא מופיע על הטופס.

ניתן לבחון מאפיין זה בזמן העיצוב על ידי יצירת פקד Image והגדרת תמונה שתוצג על ידי פקד זה, הגדרת מאפיין Stretch לערך True, ואז שינוי גודל הפקד. תרשים 19.5 מראה את אותה תמונה כפי שהיא מוצגת על ידי מספר פקדי Image. הפקד אשר מאפיין Stretch שלו מוגדר False מציג את התמונה בגודלה המקורי, ואילו שני הפקדים האחרים בהם מוגדר ערך המאפיין Stretch כ-True מציגים מספר תופעות העשויות להתרחש כתוצאה משינוי גודל הפקדים.

טיפ:

בטעינת מספר תמונות לפקד Image אחד, רצוי כי תגדיר את מאפיין Stretch של הפקד כ-True. אם לא תעשה זאת, ישתנה גודל הפקד בכל פעם שתוצג תמונה שונה. אם הצורה בה מוצגת התמונה אינה מתאימה בשל עיוות מימדיה, שקול להשתמש בפקד PictureBox במקום בפקד Image. ייתכן כי במקרים מסוימים - כגון הצגת תצלומים של אנשים - עדיף להציג תמונה חלקית מאשר תמונה שלמה אך מעוותת.



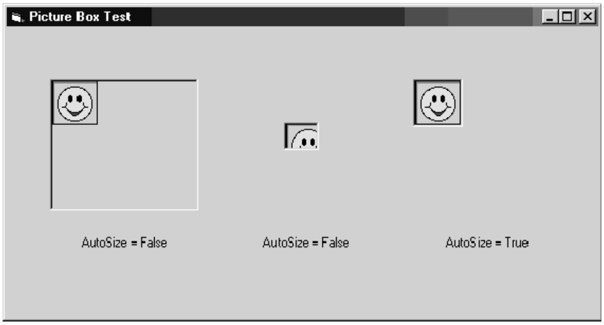
תרשים 19.5: מאפיין Stretch של פקד Image מאפשר לשנות גודל תמונה

פקד PictureBox

למרות שפקד PictureBox משתמש ביותר משאבי מערכת מאשר פקד Image, יש לו מספר תכונות נוספות שלא קיימות בפקד Image. הראשונה היא יכולתו של פקד זה לשמש כמכולה לפקדים אחרים, תכונה המאפשרת להתייחס לפקדים המצוירים בתוכו כאל קבוצה. בנוסף, ניתן להשתמש בשיטות ציור (לדוגמה Line ו-Print) כדי לצייר על תיבת התמונה.

כדי להשתמש בפקד כמכולה, ראשית הצב את הפקד על הטופס. לאחר מכן, בחר פקד מסוים מארגז הכלים וצייר אותו בתוך גבולות תיבת התמונה. כאשר תזיז את תיבת התמונה או תשנה את מאפיין Visible שלה, הדבר ישפיע גם על הפקדים הכלולים בה.

ההבדל המשמעותי ביותר בין פקד Image ופקד PictureBox הוא שפקד PictureBox אינו מאפשר לשנות את גודל התמונה. כברירת מחדל, מציג פקד PictureBox רק את אותו חלק מהתמונה אשר נכנס לגבולות הפקד. אם התמונה גדולה יותר מגודל הפקד, החלק השמאלי העליון של התמונה מוצג. אם התמונה קטנה מגודל הפקד, מושאר רווח ריק מסביב לשולי התמונה. בין אם התמונה מוצגת או לא, הרי שהיא נטענת בשלמותה לתוך הפקד, וזמינה להצגה באם גודל הפקד משתנה. הגדרת הערך True במאפיין AutoSize (גודל אוטומטי) משנה את צורת פעולתו של הפקד, כך שהוא ישנה את גודלו כדי להתאים לגודל התמונה הנוכחית. בדומה לשימוש בפקד Image, הפינה השמאלית העליונה מעוגנת במקומה, ושינוי גודל הפקד מתבצע לכיוון ימין ומטה. עם זאת, אין לבלבל בין מאפיין AutoSize של פקד PictureBox לבין מאפיין Stretch של פקד Image. פקד PictureBox שומר תמיד על יחס הצלעות המקורי של התמונה המוצגת, ללא תלות במימדי הפקד. תרשים 19.6 מראה אותה תמונה, מוצגת באמצעות שני פקדי PictureBox, אחד עם מאפיין AutoSize המוגדר False והאחר עם מאפיין AutoSize המוגדר True.



תרשים 19.6: מאפיין AutoSize מגדיר האם גודל פקד PictureBox משתנה כדי להתאים לגודל התמונה המוצגת. שים לב כי גודל התמונה עצמה אינו משתנה

הדרך הפשוטה ביותר להבין כיצד פועל מאפיין AutoSize היא ליצור יישום פשוט לניסיון. ראשית, צור פרויקט סטנדרטי עם תיבת רשימה file ועם פקד PictureBox. לאחר מכן הכנס את הקוד הבא לטופס:

```
Private Sub Form_load()
	Picture1.AutoSize = True
	File1.Path = "C:\Windows"
	File1.Pattern = "*.BMP"
End Sub
Private Sub File1_Click()
	Picture1.Picture = LoadPicture(File1.Path & "\" & File1.filename)
End Sub
```

בעוד אתה לוחץ על שם קובץ בתיבת הרשימה file, שים לב כיצד גודל תיבת התמונה מתאים עצמו לגודל התמונה שנטענה. לעיתים קרובות, מאפיין כזה יהיה בלתי רצוי שכן תיבת התמונה עלולה להסתיר פקדים אחרים. במקרה כזה ניתן להשתמש בפקד Image או לחילופין להציג את התמונה על טופס נפרד.

שיטות גרפיקה

השימוש בפקדים אינו הדרך היחידה להוספת גרפיקה ליישומים. Visual Basic מספקת גם מספר שיטות אשר יכולות לשמש לציור על טופס או בתוך פקד PictureBox. ניתן להשתמש בשיטות אלו גם עם אובייקט Printer (מדפסת) כדי לשלוח את הפלט אל המדפסת. אם משתמשים בשיטה ללא הגדרת אובייקט מסוים, פלט השיטה נשלח לטופס אשר נמצא במיקוד באותו זמן.

כדי ליצור סוגים רבים של אובייקטים גרפיים ניתן להשתמש בשיטות הבאות:

- Line - שיטה זו מציירת קו או תיבה.
- Circle - שיטה זו מציירת מעגל או אליפסה.
- PSet - שיטה זו ממקמת נקודה בודדת על אובייקט היעד.

- Point - שיטה זו מחזירה צבע של נקודה מסוימת.
- PaintPicture - שיטה זו מציירת תמונה המאוחסנת בפקד אחר, על אובייקט היעד.
- Cls - שיטה זו מנקה את אזור הפלט של אובייקט היעד.
- Print - שיטה זו ממקמת טקסט על אובייקט היעד.

באפשרותך להשתמש בכל אחת מהשיטות הללו, כדי לצייר על אובייקט. לפני שנתאר כל אחת מהשיטות, חשוב לציין כי מאפיינים מסוימים של האובייקט עליו אנו מציירים, משפיעים על הדרך בה עובדות שיטות אלו.

טבלה 19.2 מאפייני ציור המשפיעים על צורת הופעתה של גרפיקה

שם המאפיין	תפקיד
DrawMode	מגדיר כיצד משפיעים צבעי הציור והצבעים שכבר קיימים על האובייקט זה על זה
DrawStyle	מגדיר תבנית ציור
DrawWidth	מגדיר רוחב גבול של קווים ועיגולים
FillColor	מגדיר צבע המשמש למילוי מלבנים או עיגולים
FillStyle	מגדיר תבנית המשמשת למילוי מלבנים או עיגולים
ForeColor	קובע את הצבע הראשי המשמש לציור בעזרת שיטות גרפיקה, אם לא הוגדר כל צבע במהלך הקריאה לשיטה
AutoRedraw	קובע האם פלט השיטות הגרפיות מעודכן אוטומטית כאשר החלון מוסתר (רלוונטי רק לגבי Form ו-PictureBox)

קווים ומעגלים

ניתן להשתמש בשיטה **Line** לציור קווים ותיבות. כדי לצייר קו, ציין את נקודות ההתחלה והסיום של הקו ביחס למערכת צירים אשר ראשיתה בפינה השמאלית העליונה. לדוגמה, קטע הקוד הבא מצייר קו ירוק מהפינה השמאלית העליונה של הטופס ועד לפינה הימנית התחתונה:

```
Dim x2 As Single, y2 As Single
X2 = Form1.ScaleWidth
Y2 = Form1.ScaleHeight
Line (0, 0)-(x2, y2),vbGreen
```

אם תשמיט את נקודת ההתחלה של הקו, המיקום הנוכחי יהווה את נקודת ההתחלה. המיקום הנוכחי מוגדר כנקודת הסיום של הקו האחרון שצויר, ניתן להגדיר את המיקום הנוכחי באמצעות שימוש במאפייני Currentx ו-Currenty של האובייקט.

לדוגמה, הקוד הבא משמש לציור משולש על טופס. שים לב שקבוצת הקואורדינטות הראשונה מושמטת בכל קריאה לשיטה Line.

```
Private Sub
	Form1.CurrentX = 1500
	Form1.CurrentY = 750
	Line –(2000, 750), vbRed
	Line –(2000, 1250), vbBlue
	Line –(1500, 750)
End Sub
```

בהצהרה האחרונה בקטע קוד זה, השמטנו את פרמטר צבע הקו. אם לא מוגדר כל צבע, הקו יצויר בצבע המופיע במאפיין ForeColor של האובייקט. אתה יכול להוסיף את הפרמטרים B או BF אחרי פרמטר הצבע, כדי לצייר תיבות, ותיבות עם מילוי (בהתאמה). לדוגמה הפקודות הבאות מציירות שתי תיבות על טופס:

```
Line (100, 850)-(500, 1800), vbBlue, B
Line –(900, 2450),, BF
```

כאשר אתה מצייר תיבה, הקואורדינטות המועברות לשיטה Line מייצגות את הפינה השמאלית העליונה והפינה הימנית התחתונה של התיבה.

גם לשיטה **Circle** פרמטרים אפשריים רבים. תחביר השיטה בצורתו הפשוטה הוא:

```
Circle (X,Y),R
```

הפקודה מציירת מעגל ברדיוס R ואשר מרכזו נמצא בנקודה המצוינת על ידי X,Y. בדומה לשיטה Line, הדוגמה והצבע המשמשים לגבול המעגל ולמילוי, נקבעים על-פי ההגדרות של מאפייני האובייקט. להלן התחביר המלא של שיטת Circle:

```
Object.Circle [Step] (x,y), radius, [color, start, end, aspect]
```

הארגומנטים start ו-end משמשים לציור קשת במקום מעגל שלם. הערכים start ו-end הם זוויות מעל האופק ומבוטאים כרדיאנים (כדי לבצע המרה של ערך זווית ממעלות לרדיאנים, יש לחלק את ערך הזווית במספר 180). טווח הערכים של start ו-end הוא מ- -2π עד 2π. אם תשתמש בערכים שליליים עבור start ו-end, יצוירו הקווים ממרכז המעגל אל שני קצות הקשת. הקוד בדוגמה הבאה מייצר גזרה אשר נעה סביב מרכז המעגל בדומה למסך מכ״ם:

```
Private Sub Form_Click()
	Const PI = 3.14159
	Const ARCSIZE = 45
	Dim x As Integer, y As Integer, r As Integer
	Dim arcstart As Single, arcend As Single
	Dim nCount As Integer
	'Draw a circle in the middle of the form
	x = Me.ScaleWidth / 2
```

```
    y = Me.ScaleHeight / 2
    r = x / 2
    Me.DrawMode = vbCopyPen
    Me.FillStyle = vbFSTransparent
    Me.circle (x, y), r

    'Draw the sweeping radar arc
    Me.DrawMode = vbXorPen
    Me.FillColor = vbRed
    Me.FillStyle = vbSolid

    For nCount = 0 To 360
        arcstart = nCount
        arcend = nCount + ARCSIZE
        If arcend > 360 Then arcend = arcend - 360
        Me.Caption = "Arc of " & ARCSIZE & " degrees starting at " & nCount
        Circle (x, y), r, , -arcstart * PI / 180, -arcend * PI / 180
        DoEvents
        Circle (x, y), r, , -arcstart * PI / 180, -arcend * PI / 180
    Next nCount

    Me.Caption = "Done"

End Sub
```

השיטה Print

למרות שבדרך כלל אין מקשרים בין שיטת Print לגרפיקה, שיטה זו היא שיטה גרפית שכן היא "מציירת" את הטקסט על האובייקט כמו כל שיטה גרפית אחרת. ניתן להשתמש בשיטת Print בשיתוף עם שיטות גרפיקה אחרות כדי ליצור תרשימים או ציורים, או כדי להוסיף הערות לתמונות קיימות. שיטת Print כשלעצמה היא פשוטה ביותר. דוגמת הקוד הבאה מציגה שורת טקסט אחת במיקום הנוכחי של הטופס:

```
Print "This is a one-line test."
```

פלט השיטה Print נשלט על ידי מאפייני האובייקט עליו מצויר הטקסט , והם:

- CurrentX - מאפיין זה קובע את המיקום האופקי של נקודת תחילת הטקסט.
- CurrentY - מאפיין זה קובע את המיקום האנכי של נקודת תחילת הטקסט.
- Font - מאפיין זה קובע את סוג וסגנון הגופן המשמש לכתיבת הטקסט.
- ForeColor - מאפיין זה קובע את צבע הטקסט.
- FontTransparent - כאשר הטקסט מצויר על גבי טופס או תמונה, קובע מאפיין זה האם הרקע מאחורי הטקסט נראה מבעד לרווחים בטקסט.

אתה יכול להתנסות בשימוש עם שיטת Print ושיטות גרפיקה אחרות על ידי שימוש בחלון Immediate (מיידי) של Visual Basic. בצע את הפעולות הבאות:

1. צור פרויקט חדש, והוסף פקד **PictureBox**.
2. הגדר את מאפיין **AutoRedraw** של הטופס ושל תיבת הציור כ-**True**.
3. הפעל את הפרויקט ואז הקש **Ctrl+Break**.
4. הקלד פקודות המשתמשות בשיטות גרפיקה בחלון Immediate. חלון Immediate נגיש על ידי Ctrl+G או מתפריט View:

```
form1.print "Hey"
picture1.Print "Now"
```

השיטה PSet

ניתן להשתמש בשיטת PSet כדי לצייר נקודה בודדת בצבע המוגדר על ידי מאפיין ForeColor, על הטופס. גודל הנקודה תלוי בערך המאפיין DrawWidth. ככל שערך DrawWidth גדול יותר, כך יהיה גודל הנקודה גדול יותר. שיטת PSet מציירת את הנקודה במיקום המוגדר על ידי הארגומנטים שניתנים לשיטה. הקוד הבא מצייר 100 נקודות במקומות אקראיים על הטופס הנוכחי:

```
Dim I As Integer, x As Integer, y As Integer
        Me.DrawWidth = 5

        For I = 1 to 100
                'Choose random x and y
                x = Int(Me.ScaleWidth + 1) * RND
                y = Int(Me.ScaleHeight + 1) * RND
                PSet (x, y)
        Next i
```

השיטה PaintPicture

השיטה PaintPicture פועלת בדיוק כפי שמרמז שמה. היא "צובעת" תמונה מאובייקט אחד על אובייקט אחר. על ידי כיוון נכון של הארגומנטים width ו-height עבור אובייקטי המקור והיעד, באפשרותך להגדיל או להקטין את גודל תמונת המקור. קטע הקוד הבא צובע חלק מתמונה מתוך פקד PictureBox על טופס:

```
frmdest.PaintPicture picSource.Picture, 50, 50, 750, 750, 0, 0, 500, 500
```

שורת קוד זו לוקחת חלק מתמונה מפקד PictureBox בשם picSource, מגדילה וממקמת אותו על הטופס ftmDest. לשיטת PaintPicture מספר ארגומנטים:

- Source Picture - תמונת המקור, זוהי התמונה אשר חלקה או כולה מצוירת ב"אזור היעד" (Target Region) על גבי אובייקט היעד.

- Target X and Y - שני הערכים המספריים מגדירים את קואורדינטת הפינה השמאלית עליונה של אזור היעד. בדוגמה הקודמת הציור מתבצע על טופס frmDest, במרחק 50 יחידות מהקצה השמאלי ו-50 יחידות מהקצה העליון.
- Target Size -הפרמטר מגדיר רוחב וגובה של אזור היעד. אם הגודל שונה מגודל תמונת או אזור המקור, התמונה תימתח או תתכווץ כדי להתאים לאזור היעד.
- Source X and Y - זוג המספרים השלישי בפקודה מגדיר את הפינה השמאלית העליונה של אזור המקור, כלומר החלק של התמונה המועתקת.
- Source Size - פרמטר זה מגדיר את גובה ורוחב אזור המקור.

שלושת הארגומנטים הראשונים הם אלה החיוניים לפעולת השיטה PaintPicture, שאר הארגומנטים הינם אופציונליים. אם מגדירים רק את שלושת הארגומנטים הראשונים, כל תמונת המקור תועתק לאזור היעד בגודל מלא.

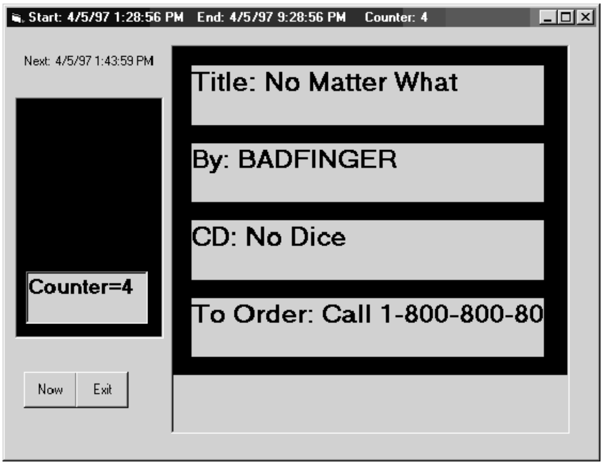
להלן מספר שימושים אפשריים לשיטת PaintPicture לצורך יצירת גרפיקה:

- לספק פונקציית הגדלה כדי לבחון מקרוב אזור מסוים של תמונה. היכולת שימושית לצורך יישום "הצג לפני הדפסה" (Print Preview) בתוכנית שלך.
- להעתיק או למחוק אזור מסוים של תמונה.
- להזיז תכולת פקד PictureBox לאובייקט Printer, לדוגמה, כדי למקם לוגו של חברה על גבי דוח.

כדי לראות דוגמה המשתמשת בשיטת PaintPicture ובמספר שיטות אחרות שהוצגו עד כה, ראה את תרשים 19.7 (זו תמונת דוגמה מיישום, היישום או הקוד אינם מצורפים לספר).

יישום זה "לוכד" מידע על שמות שירים ומבצעיהם, מתוך שידורי לווין של שירותי מוסיקה טלוויזיוניים. היישום פועל בצורה הבאה:

1. קוד המוקלד לתוך פקד Timer מפעיל את התקן לכידת הווידאו "Snappy", אשר לוכד את התמונה המוצגת על מסך טלוויזיה. תוצר לכידת הווידאו מאוחסן במאפיין Picture של פקד Snappy.
2. שיטת PaintPicture משמשת להעתקת חלקו התחתון של המסך, אשר בו מוצגים נתוני השיר, מתוך פקד Snappy לפקד PictureBox (אשר תוכנן כך שיהיה גדול יותר מפקד Snappy) הנמצא בצידו הימני של הטופס.
3. התהליך חוזר על עצמו מספר פעמים כך שבתוך פקד PicturBox נוצרת תמונה משולבת המכילה את כל המידע.
4. היישום מוסיף את ערך מונה הקלטת לפקד PictureBox באמצעות שיטת Print, והתמונה נשמרת בדיסק באמצעות שיטת SavePicture. התהליך כולו חוזר על עצמו כעבור מספר דקות עבור השיר הבא.
5. קבצי הגרפיקה שהתקבלו מספקים אמצעי לאיתור שירים על הקלטת.



תרשים 19.7: ניתן להשתמש בשיטת PaintPicture כדי לשלב מספר תמונות לתוך תמונה אחת

שיטות גרפיקה אחרות

שתי שיטות נוספות שהוזכרו בסעיף הקודם ***שיטות גרפיקה***, הן **Cls** ו-**Point**. שיטת Cls (ראשי תיבות של **Clear Screen** בדומה לפקודת DOS בעלת שם זהה) מוחקת את כל האובייקטים הגרפיים שצוירו באמצעות שיטות גרפיקה על טופס או על פקד PictureBox. שיטת Point מחזירה צבע נקודה מסוימת בתבנית RGB, כפי שניתן לראות בדוגמה הבאה:

```
Private Sub Form_load()
        'Draw a red box
        Me.AutoRedraw = True
        Line (0, 0)-(500, 500), vbRed, BF
End Sub
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, _
            Y As Single)
        If Point(X, Y) = vbRed Then
                MsgBox "The point (" & X & ", " & Y & ") is in the box"
        Else
                MsgBox "The point is not on the box"
        End if
End Sub
```

הקוד בשגרת האירוע MouseDown משתמש בשיטת Point כדי לבדוק האם המיקום של מצביע העכבר בזמן שהמשתמש לחץ על לחצן העכבר נמצא בתוך התיבה האדומה שצוירה בזמן שהטופס נטען.

לסיכום, על אף שסקירה מפורטת של אפשרויות API גרפי הוא מעבר להיקפו של פרק זה, כדאי שתשים לב כי מספר קריאות אל פונקציות API גרפיות מאפשרות לך לבצע

דברים ששיטות גרפיקה רגילות לא מאפשרות. לדוגמה, פקד ListView (תצוגת רשימה) אשר פועל כמו החלונית הימנית של Windows Explorer. במצב תצוגה Report (דוח) (או מצב Detail בתוכנת Explorer), סימון אובייקט מסוים מהרשימה, מסמן רק את העמודה השמאלית, עובדה המקשה על קריאת שורה שלמה לכל אורכה. לעומת זאת, ניתן להשתמש בקריאות API גרפי כדי לצייר קווים משלך לאורך כל העמודות. הדוגמה בתוכנית 19.1 משתמשת במספר קריאות API כדי לבצע משימה זו.

תוכנית 19.1: DRAWAPI - שימוש ב-API - גרפי להרחבת יכולות פקד ListView.

```
Private Declare Function GetDC Lib "user32" (ByVal hwnd As Long)  As Long
Private Declare Function ReleaseDC Lib "user32" (ByVal hwnd As Long, ByVal hdc _
        As Long) As Long
Private Declare Function LineTo Lib "gdi32" (ByVal hdc As Long, ByVal X As Long, _
        ByVal Y As Long) As Long
Private Declare Function MoveToEx Lib "gdi32" (ByVal hdc As Long, ByVal X As Long, _
        ByVal Y As Long, lpPoint As POINTAPI) As Long

Private Type POINTAPI
    X As Long
    Y As Long
End Type

Private Sub Form_Load()

    'This code adds 100 items to a listview control

    Dim objTemp As ListItem
    Dim i As Integer
    Me.ScaleMode = 3

    With Me.ListView1
        .View = lvwReport
        .ColumnHeaders.Add,, "Column1"
        .ColumnHeaders.Add,, "Column2"
        For i = 1 To 100
            Set objTemp =.ListItems.Add(, , "Item " & i)
            objTemp.SubItems(1) = "Top " & objTemp.Top
            .Refresh
        Next I
    End With

End Sub
```

```
Private Sub ListView1_ItemClick(ByVal Item As ComctlLib.ListItem)
	Dim hdc As Long
	Dim pointx As POINTAPI
	Dim lRetVal As Long
	Dim nHeight As Integer
	Dim lTemp As Long
	'The form's ScaleMode property MUST be
	'set to pixels for this to work properly.

	nHeight = Me.TextHeight("X")

	With ListView1
		.Refresh
		hdc = GetDC(.hwnd)
		lTemp = MoveToEx(hdc, 0, Item.Top, pointx)
		lTemp = LineTo(hdc,.Width, Item.Top)
		lTemp = MoveToEx(hdc, 0, Item.Top + nHeight, pointx)
		lTemp = LineTo(hdc,.Width, Item.Top + nHeight)
		lTemp = ReleaseDC(.hwnd, hdc)
	End With

End Sub
```

הקוד בתוכנית 19.1 מדגים את אחד העקרונות הבסיסיים בשימוש בקריאות אל API גרפי: הקשר ההתקן (Device Context). הקשר התקן יכול לשמש קריאות אחרות אל API גרפי, כדי לצייר על אובייקטים. הדוגמה משתמשת בקריאה GetDC כדי לקבל את הקשר ההתקן של פקד ListView, ואז היא מבצעת קריאה אל LineTo כדי לצייר על פקד ListView.

עבודה עם טקסט וגופנים

כמעט כל תוכנית שתכתוב ב- Visual Basic תדרוש ממך להשתמש בטקסט. בפרקים קודמים למדת על הצגת טקסט בתוויות ובתיבות טקסט. כמו כן למדת על פונקציות מחרוזת, המשמשות לעיבוד טקסט ברמת קוד התוכנית. פרק זה ירחיב בנושא השימוש בטקסט ויראה לך כיצד להציג טקסט בצורה האינטואיטיבית ביותר עבור משתמשי התוכניות שלך.

התנהגות תיבת טקסט

בפרק 4, **שימוש בפקדי ברירת המחדל של Visual Basic**, הוצגה בפניך תיבת הטקסט. אתה כבר יודע כי מאפיין Text משמש לאחסון ואחזור מידע. בנוסף למאפיינים הרגילים שנדונו בפרק זה, יש מספר מאפיינים נוספים ההופכים את פקד TextBox (תיבת טקסט) לגמיש אף יותר:

- Locked (נעול) - מאפיין זה מונע מהמשתמש להקליד נתונים לתיבת הטקסט.
- MaxLength (אורך מקסימלי) - מאפיין זה מגביל את מספר התווים שתיבת הטקסט מקבלת.
- PasswordChar (תו סיסמה) - מאפיין זה גורם לתיבת הטקסט להסתיר את המידע המוקלד על ידי המשתמש.
- SelLength (אורך בחירה), SelStart (תחילת בחירה), SelText (טקסט נבחר) - מאפיינים אלה מאפשרים למשתמש לשנות רק את החלק המסומן של הטקסט.

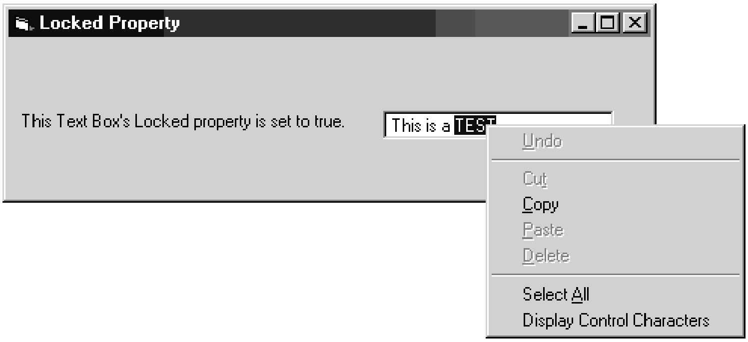
נעילת גישה למשתמשים

ראשית, נדון במאפיין Locked, אשר מאפשר לך להשתמש בתיבת טקסט לצורכי תצוגה בלבד. השאלה המתבקשת היא, מדוע לעשות כך במקום להשתמש בפקד Label לצורך התצוגה? התשובה היא, שלמרות שתיבת טקסט "נעולה" אינה מאפשרת למשתמש להכניס מידע, היא מאפשרת לו לגלול, לבחור ולהעתיק קטעי טקסט. כמובן שקוד היישום עצמו יכול תמיד לשנות את תוכן תיבת הטקסט, בין אם היא נעולה או לא. כדי לנעול תיבת טקסט, פשוט הגדר את מאפיין Locked של התיבה True, כפי שניתן לראות בדוגמה הבאה:

```
'LOCKS A SINGLE TEXT BOX
txtTest.Locked = True
txtTest.Text = "You can't edit this!"

'LOCKS EVERY TEXT BOX IN THE FORM
Dim objControl As Control
For Each objControl In Me.Controls
	If TypeOf objControl Is TextBox then objControl.Locked = True
Next objControl
```

מנקודת מבט ממשק המשתמש, מאפיין Locked מאפשר לך להשתמש באותו מסך לשם הכנסת וחיפוש נתונים. נעילת תיבת טקסט מונעת מאנשים בלתי מוסמכים לשנות מידע, כפי שמוצג בתרשים 19.8.



תרשים 19.8: השימוש במאפיין Locked מונע עריכה בלתי מכוונת של טקסט. שים לב כי בתפריט המשנה (הנבנה אוטומטית על ידי Windows) רק האפשרות **Copy** זמינה, אם תיבת הטקסט נעולה



הערה:

אין לבלבל בין מאפיין Locked למאפיין Enabled (פעיל). מאפיין Locked אינו יוצר את אפקט grayed-out (סימון באפור) (כלומר התיבה פעילה ונגישה אך אי אפשר לערוך את תוכנה). בנוסף, מאפיין Locked מאפשר לבחור ולהעתיק טקסט מתוך התיבה, בעוד מאפיין Enabled אינו מאפשר זאת.

מאפיין MaxLength

כאשר אתה מעצב טופס להזנת נתונים, אחת מהמשימות אשר אתה חייב לבצע היא בדיקת **תקפות הנתונים** (Data Validation). סוג אחד של בדיקת תקפות הנתונים הוא הבטחת התאמתו של המידע המוקלד לגודל השדה המיועד בבסיס הנתונים. למרות שבדיקה כזו יכולה להתבצע בקוד עצמו על ידי שימוש בפונקציה Len, ניתן להימנע מהצורך בקוד הנוסף על ידי שימוש במאפיין MaxLength של פקד TextBox. מאפיין Maxlength מאפשר להגדיר את מספר התווים המקסימלי שניתן להכניס לתיבת טקסט, ללא קשר לגודל התיבה על הטופס.

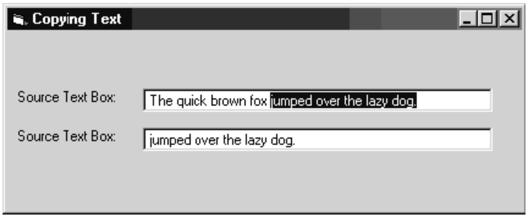
מאפיין PasswordChar

אם אתה משתמש בתיבת טקסט כחלק מתהליך כניסה לרשת (Login) תרצה בוודאי להסתיר את הסיסמאות המוקלדות על ידי המשתמשים. כדי לבצע זאת, פשוט הכנס תו כלשהוא למאפיין PasswordChar של תיבת הטקסט. מאפיין זה משנה את התנהגות תצוגת תיבת הטקסט, כך שהתו שהגדרת כי הוא PasswordChar מוצג בתיבה במקום התווים במאפיין Text. ייתכן כי ראית אפקט זה פעמים רבות כאשר ביצעת Login אל Windows.

שים לב כי מאפיין Text עדיין מכיל את התווים המקוריים שהוקלדו, וקוד התוכנית רואה את הטקסט ה"אמיתי". למרות שניתן להשתמש בכל תו למטרה זו, נהוג להשתמש בתו הכוכבית (*) להסתרת טקסט.

עריכה בתיבת טקסט

תיבת טקסט סטנדרטית מאפשרת למשתמש לסמן טקסט באמצעות העכבר, על ידי הקשת Ctrl+Shift, או הזזת הסמן באמצעות החיצים. אז, יכול קוד התוכנית לעבד את הטקסט הנבחר באמצעות מאפייני SelText, SelLength ו-SelStart. ניתן להשתמש במאפיינים אלה מתוך התוכנית, כדי לעבוד עם קטעי טקסט שנבחרו. הבט בדוגמה המוצגת בתרשים 19.9. במקרה זה מאפיין SelText יכיל רק את המילים "jumped over the lazy dog". מאפיין SelLength יכיל את הערך השלם 25, שהוא אורכה של מחרוזת תווים זו. ומאפיין SelStart יכיל את הערך 20, כלומר הטקסט הנבחר מתחיל בתו מספר 20 שבתיבת הטקסט.



תרשים 19.9: המשתמש יכול להעתיק טקסט מתיבה אחת, ולהדביקו בתוך תיבה אחרת

בנוסף על בדיקת איזה חלק מהטקסט נבחר, ניתן גם להגדיר את ערכי מאפיינים אלה מתוך הקוד וכך לשנות את התחום הנבחר. בכל פעם שמגדירים את ערך המאפיין SelStart יש להגדיר גם את מאפיין SelLength כדי לבחור מספר תווים. כדי לסמן את שלושת התווים הראשונים, ניתן להשתמש בקטע הקוד הבא:

```
txtTest.SelStart = 0
txtTest.SelLength = 3
```

ניתן לשנות את ערך המאפיין SelLength מספר פעמים. פעולה זו גורמת לקטע הנבחר לגדול או לקטון, וכן לעדכון אוטומטי של מאפיין SelText. הגדרת ערך המאפיין SelText מתוך הקוד גורמת לקטע הטקסט הנבחר הנוכחי להיות מוחלף על ידי מחרוזת התווים החדשה, לדוגמה:

```
txtTest.SelTest = "jumped onto incoming traffic."
```

אחד השימושים למאפיינים אלה הוא סימון כל תכולת תיבת הטקסט. נניח לדוגמה כי מוצגות מספר תיבות טקסט בו-זמנית, והמשתמש אמור לערוך את תוכנן. נרצה לגרום לכך שבזמן שהמשתמש מקיש Tab כדי לעבור בין התיבות, תסומן כל תכולת תיבת הטקסט, כך שהמשתמש יוכל להקליד את הקלט מבלי למחוק את הטקסט שמופיע בתיבה. בדוגמה הבאה מיושם רעיון זה בשגרת האירוע GotFocus של תיבת טקסט:

```
Private Sub txtSelect_GotFocus()
	txtSelect.SelStart = 0
	txtSelect.SelLength = Len(txtSelect.Text)
End Sub
```

ניתן ליישם מאפיין זו גם על ידי שימוש בפקודת SendKeys (שלח מקשים) כדי לשלוח את צירוף המקשים הנדרש לסימון כל תכולת תיבת הטקסט:

```
Private Sub txtSelect_GotFocus()
    SendKeys ("{HOME}+{END}")
End Sub
```

עבודה עם גופנים וצבעים

למרות שתמונות וציורים מוסיפים רושם ויזואלי ניכר ליישומים שלך, עיקר רוב התוכניות יהיה כנראה שדות הטקסט להכנסת והצגת נתונים. לעיתים הוספת תמונה לצורך הדגשת טקסט מסוים אינה אפשרית ואף אינה מועילה. במקום זאת יש להשתמש בגופנים ובצבעים הנכונים כדי ליצור את הרושם הרצוי.

אובייקט Font

אם השתמשת במעבד תמלילים בעבר, אתה מכיר בוודאי את המושג גופן (font)**.** במהלך העיצוב ניתן להגדיר גופנים לפקדים ולטפסים על ידי שימוש בחלון המאפיינים. לאחר הגדרת גופן לשימוש הטופס, ישמש גופן זה להצגת כל הפקדים שיצוירו על הטופס.

ייתכן כי אתה זוכר מפרקים קודמים, כי מאפיין Font של אובייקט הוא למעשה אובייקט בעצמו בעל מאפיינים משלו. להלן מאפייני האובייקט Font המשמשים לשליטה על צורת הופעת הגופנים בתוכנית:

- Name (שם) - מאפיין שם לאחד הגופנים המותקנים במערכת, לדוגמה: "Arial" או "Times New Roman".
- Bold (מודגש) - מאפיין True/False אשר שולט באפקט ההדגשה של הגופן. אותיות בעלות מאפיין זה נראות **כהות ועבות** יותר מאותיות רגילות.
- Italic (נטוי) - מאפיין True/False אשר קובע האם האותיות *נטויות*.
- Underline (קו תחתי) - מאפיין True/False הקובע האם הטקסט מופיע <u>עם קו תחתון מתחת לכל תו</u>.
- Size (גודל) - מאפיין זה שולט בגודל הגופן, ערך המאפיין מוגדר בנקודות (Point). נקודה אחת שווה ל-1/72 אינטש, כלומר אותיות בגודל 72 נקודות הן בגובה אינטש אחד בערך.
- Strikethrough (קו חוצה) - מאפיין True/False הקובע האם יצויר קו דק במרכז ~~הטקסט~~ לכל אורכו.
- Weight (עובי) - שולט בעובי הקו המשמש לכתיבת הטקסט. שני הערכים האפשריים למאפיין זה הם 400 ו-700, המשמשים לכתיבת טקסט רגיל ומודגש בהתאמה.
- Charset (סט תווים) - קובע איזו קבוצת תווים תשמש לכתיבה (Standard, Extended DOS, Japanese). מאפיין זה מוסבר ביתר פירוט בקובץ Help.

❖ כפי שכבר שמת לב בוודאי, כדי להגדיר את השימוש בגופן מסוים עליך לדעת את שמו. ייתכן כי במערכות שונות יותקנו גופנים שונים. לאובייקטים Screen (מסך) ו-Printer יש אוספי Fonts המפרטים את סוגי הגופנים הקיימים. ניתן לכתוב תוכנית פשוטה אשר תבנה רשימת כל הגופנים המותקנים במערכת. התחל פרויקט חדש, והקלד את הקוד הבא לשגרת האירוע Click של הטופס. לאחר הפעלת התוכנית, לחיצה שמאלית על העכבר תגרום להדפסת שמות כל הגופנים המותקנים על גבי הטופס:

```
Dim nCount As Integer
Me.AutoRedraw = True
For nCount = 0 To Screen.FontCount – 1
	Me.Font.Name = Screen.Fonts(nCount)
	Me.Print Screen.Fonts(nCount)
Next nCount
```

כמו כן, שים לב כי רוב הגופנים מכילים יותר תווים מאלה שניתן להפיק באמצעות המקלדת. לדוגמה, גופן MS Sans Serif מכיל תווים מיוחדים להצגת שברים ואותיות מוטעמות. ניתן להציג תווים אלה על ידי שימוש בפונקציית Chr$. הקוד הבא מציג את כל 256 התווים הקיימים (כאשר רבים מהם אינם ניתנים להצגה כלל) על הטופס:

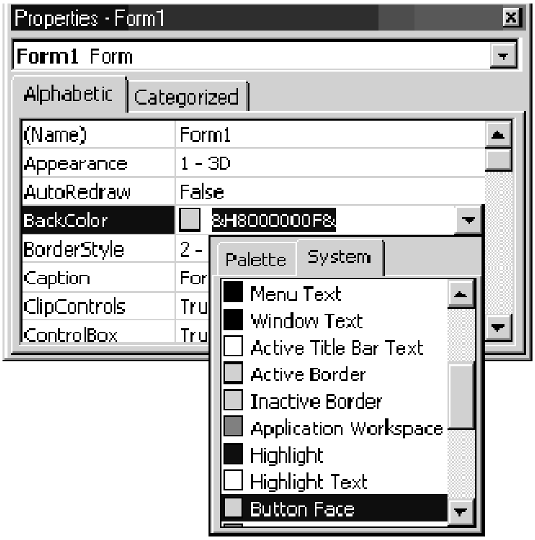
```
Dim i As Integer
For i = 0 To 255
	Me.Print Chr$(i); " ";
	If Me.CurrentX >= Me.Width Then Me.Print
Next i
```

הוספת צבע

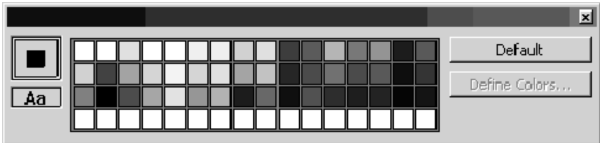
טופס המעוצב בעזרת רכיבים אפורים בלבד נוטה להיות משעמם. ניתן להקצות צבעים לטופס ולפקדים בקלות על ידי שימוש במאפיינים ForeColor ו-BackColor. כדי להגדיר מאפיינים אלה בזמן העיצוב, ניתן להשתמש ברשימת הצבעים (Color list) או בלוח הצבעים (Color Palette). כדי לראות את רשימת הצבעים, פתח את חלון המאפיינים של טופס או פקד כלשהו, ולחץ על המאפיין ForeColor או BackColor. רשימת הצבעים מוצגת בתרשים 19.10.

שים לב כי רשימת הצבעים מורכבת משני חלקים: Palette (לוח) ו-System (מערכת). Palette מאפשר לך לבחור צבעים ספציפיים, בעוד System מציג בפניך את הצבעים בסכימת הצבעים הנוכחית של Windows. אם תשתמש בצבעי System, כל שינוי שתבצע בהגדרת צבעי Windows בלוח הבקרה, Display (תצוגה), יתבטא גם בצבעי התוכנית שלך. יש לקחת לתשומת לב עובדה זו, כאשר שוקלים הפצת תוכנה.

לוח הצבעים, המוצג בתרשים 19.11, מאפשר לך לשנות את צבעי הקידמה והרקע באותו זמן. כדי לראות את לוח הצבעים, סמן פקד או טופס ואז לחץ על **Color Palette** מתוך תפריט **View**.



תרשים 19.10: ניתן להגדיר את מאפייני ForeColor ו-BackColor מרשימת הצבעים



תרשים 19.11: תיבת לוח הצבעים היא אמצעי נוסף לשינוי צבעים בזמן העיצוב

שנה צבעים בזמן העיצוב, והבט בערכי המאפיינים ForeColor ו-BackColor. תוכל לראות שהערכים הם למעשה מספרים. לדוגמה צבע כחול מוצג על ידי המספר &HFF0000& בבסיס הקסדצימלי או 16711680 בבסיס דצימלי. ב- Visual Basic מוגדרים גם מספר קבועים מובנים לצבעים מסוימים, ראה טבלה 19.3. כדי להגדיר צבעים של אובייקטים בתוכנית שלך מתוך הקוד, פשוט הצב את המספרים הנכונים במאפייני ForeColor ו-BackColor כפי שמדגים קטע הקוד הבא:

```
Form1.ForeColor = vbYellow
Form1.BackColor = vbBlue
```

הערה:

קיימים גם קבועים לצבעי System. לדוגמה, הפקודה הבאה מגדירה את צבע הטופס לצבע שולחן העבודה של Windows:

```
Form1.BackColor = vbDesktop
```

טבלה 19.3: קבועי הצבעים המובנים ב- Visual Basic

שם הקבוע	ערך מספרי (בבסיס דצימלי)
vbBlack	0
vbRed	255
vbGreen	65280
vbYellow	65535
vbBlue	16711680
vbMagenta	16711935
vbCyan	16776960
vbWhite	16777215

הסיבה שקבועי הצבעים מיוצגים על ידי מספרים לא רגילים אלה, היא שכל מספר מייצג את רמות שלושת צבעי היסוד: אדום, ירוק וכחול המרכיבים את הצבע הסופי. כדי ליצור צבע מותאם, ובידיעה מהי רמת כל אחד מצבעי היסוד בצבע זה, היעזר בפונקציה RGB כדי לחשב את ערך הצבע. ערך כל צבע יסוד נע בין 0 ל-255. לדוגמה, קטע הקוד הבא מחשב את ערך הצבע האפור (רמות שוות של אדום, ירוק וכחול):

```
Dim lNewColor As Long
lNewColor = RGB(192, 192, 192)
Form1.BackColor = lNewColor
```

הערה:

דרך פשוטה לבדוק את ערכו של צבע מסוים לצורך שימוש בקוד היא לבחור צבע זה מתוך רשימת הצבעים ולראות מהו הערך המספרי המוצג בחלון המאפיינים.

פונקציית צבע שימושית נוספת היא QBColor. בגרסאות ישנות יותר של Basic, כמו QuickBasic ו-GWBasic ב- MS-DOS, צבעים היו מיוצגים על ידי מספרים שלמים עוקבים. פונקציית QBColor מחזירה את ערך RGB בבסיס הקסדצימלי של צבע מסוים. יכולת זו שימושית אם ברצונך לייצר צבעים אקראיים, כמוצג בדוגמה הבאה:

```
Private Sub Form_Click()

    Dim lRGBColor As Long
    Dim nIntColor As Integer

    'Choose random number between 0 and 15
    nIntColor = Int(16 * Rnd)
    lRGBColor = QBColor(nIntColor)
    Me.BackColor = lRGBColor
End Sub
```

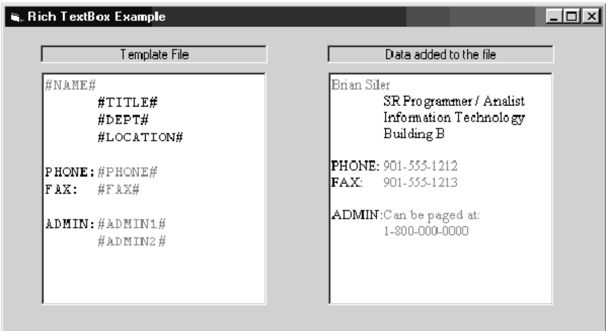
אם מקלידים קוד זה לשגרת אירוע Click של טופס, תגרום כל לחיצת עכבר על גבי טופס זה לשינוי צבע הרקע שלו.

גופנים, צבעים ופקד RichTextBox

תיבת הטקסט טובה מאוד לשימושים כלליים, אך פקד RichTextBox מאפשר יותר שליטה על הדרך בה מוצג טקסט. העוצמה של פקד RichTextBox, אשר הוצג כבר בפרק 6 **שליטה נוספת למשתמש: תפריטים וסרגלי כלים**, נובעת מיכולתו להציג מספר פורמטים וצבעים שונים באותו אזור טקסט.

שימוש אפשרי אחד לפקד זה הוא הצגת מידע מפורט מתוך בסיס נתונים בתבנית מסודרת, כפי שניתן לראות בתוכנית המוצגת בתרשים 19.12. כאשר משתמש לוחץ על שמו של אחד העובדים, תוארו ומידע אחר הקשור אליו מוצג בתיבת הטקסט העשירה בצידו הימני של הטופס.

למרות שהמידע מוצג בתבנית מהודרת, ניתן להשיג תוצאה זו בהשקעת תכנות מינימלית. הצעד הראשון הוא יצירת תבנית לתצוגה המפורטת הרצויה באמצעות מעבד תמלילים כמו WordPad או Microsoft Word. עצב את המסמך באותה צורה בה היית רוצה שהנתונים שלך יוצגו, אך אל תקליד כל נתון לתוך הקובץ אלא השאר "שומרי מקום" במקומות בהם יוכנסו הנתונים. דוגמה לתבנית כזו ניתן לראות בתרשים 19.12. שים לב כי התו # משמש לעטיפת כל שומרי המקום כדי לא לבלבלם עם נתונים ממשיים.



תרשים 19.12: יש ליצור את התבנית המעוצבת מסוג RTF פעם אחת בלבד, ולאחר מכן ניתן להשתמש בה בקלות להצגת נתונים

כדי להציג נתונים בתיבת הטקסט העשיר, עליך לטעון את התבנית ולמצוא ולהחליף את כל שומרי המקום. דוגמת הקוד הבאה משתמשת בפונקציה המוגדרת על ידי המשתמש בשם RepRTBfield אשר מוצאת שומרי מקום ומחליפה אותם עם נתונים:

```
Sub RepRTBfield(sField As Integer, sValue As String)
	rtbInfo.Find "#" & sField & "#"
	rtbInfo.SelText = sValue
End Sub
Private Sub cmdDisplay_Click()
	rtbInfo.LoadFile (App.Path & "\info.rtf")
	RepRTBfield "NAME", "Brian Siler"
	RepRTBfield "DEPT", "Acounting"
	RepRTBfield "TITLE", "CIA"
End Sub
```

למרות שהקוד בדוגמה זו הוא פשוט, אתה יכול להרחיב אותו בקלות לכדי כלי עיצוב רב עוצמה. שיטת Find של תיבת טקסט עשיר מאפשרת לך לבצע החלפה של קודים מוגדרים משלך בתבנית RTF, במקום לבחור ולעצב כל פיסת טקסט נפרדת באמצעות קוד Visual Basic.

מכאן...

למרות שלא ניתן להציע סדרת צעדים בדוקה אשר תבטיח פיתוח ממשק משתמש מוצלח, חשוב לדעת כי ממשק משתמש גרוע יגרום לכך שאיש לא ירצה להשתמש בתוכניות שתכתוב. עם זאת, בקצב המהיר בו מתפתח עולם המחשוב, הגדרת ממשק משתמש טוב תמשיך להשתנות. ניקח כדוגמה את תהליך כיוונון השעון של מכשיר וידאו. כיוונון שעונים של מכשירי וידאו ישנים נעשה באמצעות לחצנים ומפסקים, אך שיטה זו הוחלפה במהירות בשיטה מתקדמת יותר של תצוגה על מסך הטלוויזיה. כיום, שידורים אלחוטיים הופכים את כל התהליך לאוטומטי לחלוטין בדגמים מסוימים. כמו מכשיר הווידאו, ממשק המשתמש לתוכניות Visual Basic שלך יתפתח במשך הזמן, בעוד תעשיית התוכנה תקבע סטנדרטים חדשים, ואתה תלמד כיצד לעמוד בצפיות המשתמשים שלך.

כדי ללמוד עוד על ההיבטים הוויזואליים של עיצוב ממשק משתמש, עיין בפרקים הבאים:

- כדי ללמוד על פקדים המשמשים לעיצוב תוכניות Visual Basic, ראה פרק 4 - **שימוש בפקדי ברירת המחדל של Visual Basic** ופרק 12 - **הפקדים המשותפים של Microsoft**.
- לטיפים כלליים לגבי עיצוב הממשק, ראה פרק 18 - **תכנון ממשק נכון**.
- כדי ללמוד כיצד לעצב יישום לשימוש ב-web, ראה פרק 31 - **מסמכי ActiveX**.

חלק 5

נושאים מתקדמים בתכנות



20

גישה ל-API של Windows

מה בפרק?

- הבנת API של Windows
- שימוש ב-API של Windows ב- Visual Basic
- קריאות API שימושיות

API (ממשק פיתוח היישום - Application Programming Interface) של Windows הוא קבוצת פונקציות שמערכת ההפעלה Windows חושפת בפני המשתמש. ניתן לקרוא לפונקציות אלו, אשר מהוות למעשה חלק ממערכת ההפעלה Windows, מתוך Visual Basic כדי לבצע משימות אשר לא ניתן לבצען בקוד Visual Basic סטנדרטי. לדוגמה, לא קיימת פונקציית Visual Basic המבצעת אתחול (Reboot) למחשב. עם זאת ניתן לבצע פעולה זו באמצעות שימוש בפונקציית API של Windows. ניתן לחשוב על רבות מהפונקציות הכלולות ב- Visual Basic כעל "עטיפות" לפונקציות מתוך API של Windows. השימוש בפונקציות מתוך API של Windows הוא מסובך יותר מאשר כתיבת פונקציות מותאמות אישית על ידי המשתמש. לכן נהוג לכתוב עטיפות לפונקציות API בהן רוצים להשתמש, ולקרוא להן באמצעות פונקציות אלו. למרות שלא ניתן לראות בפרק זה מדריך מעמיק לנושא זה, הוא מעניק בסיס איתן להכרת השימושים המצויים לפונקציות API.

הבנת API של Windows

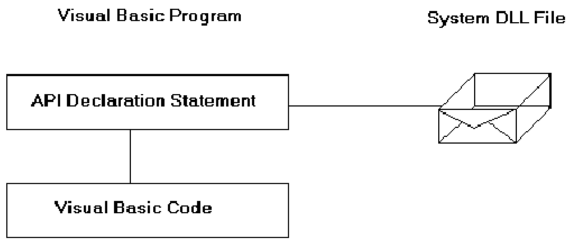
מנקודת מבטו של מפתח Visual Basic, ניתן לחשוב על פונקציות API כעל פונקציות Visual Basic "נורמליות". יש להן פרמטרים לקלט ולפלט, ולפעמים הן מחזירות ערך לתוכנית שקראה להן. אך פונקציות API הן פונקציות שעברו תהליך הידור, ומאוחסנות בקבצים נפרדים הנקראים ספריות קישור דינמיות - **DLL** (Dynamic Link Library).

כדי להשתמש בפונקציות אלו עליך להוסיף מספר שורות קוד המגדירות את הפונקציה החיצונית עבור Visual Basic. במילים אחרות, כדי להשתמש בפונקציית API עליך **להכריז** (Declare) עליה תחילה. הכרזות API מוכנסות באזור General Declarations (הכרזות כלליות) של מודול. ממש כפי שמכריזים על משתנים, יש להכריז על פונקציות API כדי שקוד התוכנית יוכל להשתמש בהן. וכפי שמשתנה שהוכרז מספק קישור לאזור בזיכרון בו מאוחסן ערכו, פונקציית API שהוכרזה מספקת קישור לקובץ DLL חיצוני.

הערה:

המקום הרגיל להכרזות API הוא במודול, אך ניתן להוסיף הכרזות אלו גם לטפסים ולמחלקות על ידי הוספת מילת המפתח Private (פרטי) לפני ההכרזה.

פקודת Declare כוללת רשימה של כל הפרמטרים שמקבלת פונקציית API עליה מכריזים, קובץ DLL בו נמצאת הפונקציה, וסוגי הנתונים המוחזרים על ידי הפונקציה. שלא כמו פונקציות Visual Basic רגילות, הכרזת פונקציית API לא כוללת כל קוד. במקום זאת, פקודת ההכרזה כוללת הפניה לקובץ DLL המכיל את הפונקציה. מבנה זה מתואר בתרשים 20.1.



תרשים 20.1: פונקציות API מאוחסנות בקבצי DLL של מערכת ההפעלה. הכרזת API מאפשרת לקוד Visual Basic "לראות" פונקציות אלו

הערה:

הצהרות Declare רבות כוללות פרמטר בשם Alias. ה**כינוי** (Alias) מגדיר את שמה "האמיתי" של פונקציית API כפי שהוא מאוחסן בקובץ DLL, שם זה יכול להיות שונה מהשם בו אתה מעוניין לקרוא לפונקציה זו בתוכנית שלך. לדוגמה, DLL בשם kernel32 כולל פונקציה בשם _lopen, אך שם זה אינו קביל כשם פונקציה ב- Visual Basic. במקרה כזה הדרך הנכונה לבצע הכרזת API היא:

```
Declare Function lopen Lib "kernel32" Alias "_lopen" (ByVal lpPathName As String,  _
        ByVal iReadWrite As Long) As Long
```

Visual Basic רואה את הפונקציה בשם lopen אך היא יודעת על פי פרמטר Alias שנכלל בפקודת Declare להעביר כל קריאה המתבצעת לפונקציה זו, לפונקציה אחרת בשם _lopen הנמצאת בתוך DLL בשם kernel32.

הבה נדגים את השימוש בפונקציית API באמצעות הדוגמה שהבאנו בהקדמה לפרק זה, אתחול המערכת. ראשית, התחל פרויקט Standard EXE חדש, והוסף מודול חדש על ידי לחיצה על Add Module מתוך תפריט Project. לאחר מכן, עבור לאזור ההגדרות של המודול, והקלד את פקודת Declare הבאה בשורה אחת:

```
Declare Function ExitWindowsEx Lib "user32" (ByVal uFlags As Long, _
        ByVal dwReserved As Long) As Long
```

הערה:

שים לב לקו התחתון ("_" underscore) המופיע בסוף השורה הראשונה. ניתן לחלק פקודת Declare למספר שורות, על ידי שימוש ב"תו המשכיות השורה" (_).

שים לב כי הפקודה Declare מכילה את שם הפונקציה (ExitWindowsEx), את שם ה-DLL בו נמצאת הפונקציה (user32), ואת רשימת הפרמטרים שהפונקציה מקבלת. לאחר הקלדת קטע קוד זה, ניתן לקרוא לפונקציה ExitWindowsEX מתוך קוד Visual Basic בדומה לקריאה לפונקציית Visual Basic רגילה.

יש לזכור, כי לפני הקריאה לפונקציה צריך להגדיר ערכים חוקיים לשני הפרמטרים לפונקציה. הקבועים הבאים הם ערכים אפשריים עבור הפרמטר הראשון, uFlags:

```
Public Const EWX_FORCE = 4
Public Const EWX_LOGOFF = 0
Public Const EWX_REBOOT = 2
Public Const EWX_SHUTDOWN = 1
```

אם תבחן את הקבועים שהוגדרו בקטע הקוד הקודם, ייתכן כי תשים לב שהם מתאימים לאופציות העומדות לרשותך כשאתה סוגר את Windows. כל אחד מהקבועים המועברים לפרמטר uFlags גורם לפונקציה לפעול בצורה שונה, בדומה למאפיין Flags של פקד CommonDialog.

הפרמטר השני, dwReserved, הוא פרמטר טיפוסי לפונקציות API רבות. זהו פרמטר שמור אשר לא נועד לשימוש, לכן ניתן להגדיר את ערכו 0.

הערה:

קבועים המשמשים עם פונקציות API, מתנהגים כמו קבועי Visual Basic רגילים. כדי להקל על קריאת הקוד, מומלץ למקם את הקבועים המשמשים את פונקציות API באותו מודול בו מוכרזות הפונקציות.

לאחר הכרזת הפונקציה והגדרת הקבועים, ביצוע הקריאה מתוך הקוד הוא תהליך פשוט למדי. לדוגמה, הקלד את שורות הקוד הבאות לשגרת Form_Load כדי לאתחל את המחשב:

```
Dim lRetVal As Long
lRetVal = ExitWindowsEX(EWX_REBOOT, 0)
```

הפעלת קטע קוד זה תגרום למערכת ההפעלה Windows להתחיל כיבוי של המערכת, ולשאול את המשתמש האם ברצונו לשמור קבצים פתוחים.

API של Windows ב- Visual Basic

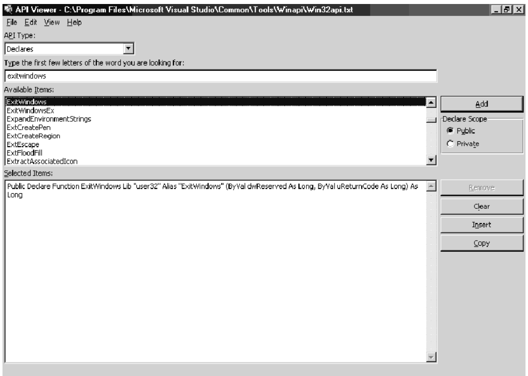
API של Windows נקרא לפעמים API **Win32**, כאשר המספר 32 מתייחס לשימוש בפונקציות 32Bit. API שהכיל פונקציות 16Bit, שימש בגירסה 3.11 של Windows ובגרסאות קודמות של Visual Basic. למרות שמערכות ההפעלה Windows 95 ו-Windows NT מספקות תאימות לאחור עבור יישומים אשר עברו הידור עם קריאות לפונקציות 16Bit, לא ניתן להכריז על פונקציות 16Bit מתוך Visual Basic 6.0. חשוב לזכור נקודה זו כאשר מבצעים שדרוג של פרויקטים ישנים.

API Viewer

הכרזות פונקציות API, הקבועים וסוגי הפרמטרים שהן מקבלות, רשומים בקובץ בשם WIN32API.TXT. קובץ זה מותקן בתיקיה בשם WINAPI תחת התיקיה בה הותקנה Visual Basic. כדי להשתמש בפונקציית API בתוכנית שלך, פשוט העתק והדבק את המידע הדרוש לך מתוך קובץ זה. בנוסף לקוד, תמצא בקובץ זה גם הערות שיעזרו לך להבין את תפקיד הפונקציות השונות.

ערכת הפיתוח של Visual Basic כוללת כלי בשם API Text Viewer (מציג טקסט API), אשר נועד להאיץ את פעולת הוספת פונקציות API. כלי זה, המוצג בתרשים 20.2, מארגן את פונקציות API של Windows בסדר אלפביתי, ומאפשר למפתח להעתיק מספר פונקציות API ללוח.

כדי להשתמש ב- API Viewer, לחץ על הסמל API Text Viewer בקבוצת התוכניות Visual Basic. לאחר מכן, לחץ על Load Text File מתוך תפריט File, ובחר WIN32API.TXT (שים לב כי ניתנת לך האפשרות להמיר את קובץ הטקסט WIN32API.TXT לקובץ בסיס נתונים לשם טעינה מהירה יותר). לאחר ש- API Viewer מעבד את קובץ הטקסט, מוצגת בפניך רשימה של הכרזות API בתיבת Available Items.



תרשים 20.2: API Viewer מאפשר לך להעתיק כל שילוב סוגים, הכרזות וקבועים

נסה זאת בעצמך על ידי העתקת פונקציית GetDiskFreeSpace ללוח. הדרך המהירה ביותר לאתר פונקציה מסוימת היא להקליד את האותיות הראשונות של שמה, וכך "לקפוץ" ישירות למיקומה ברשימה.

לפני שנמשיך, שים לב ליכולת חדשה של API Viewer: האפשרות לבחור הכרזות Public (ציבוריות) או Private (פרטיות). מה שעושה אפשרות זו למעשה, היא להוסיף את המילים **Private** או **Public** להכרזות. באפשרותך עדיין לערוך את הכרזות הפונקציות במועד מאוחר יותר. הכרזת פונקציית API כציבורית מאפשרת למקם אותה בקוד של מודול מסוים, ולקרוא לה מתוך טפסים או מודולים אחרים באותו פרויקט. מצד שני, אם ברצונך למקם הכרזת פונקציית API בקוד של טופס, יש להגדיר אותה כפרטית. מאחר והדוגמה שלנו משתמשת בטופס, לחץ על הלחצן Private.

לחץ לחיצה כפולה על GetDiskFreeSpace ותוכל לראות כי הפונקציה מופיעה בתיבת Selected Items (פריטים נבחרים). שים לב כי ניתן ללחוץ לחיצות כפולות על פונקציות נוספות כדי להוסיפן לתיבת Selected Items. ניתן להציג את הקבועים והסוגים בהם API משתמש, על ידי סימון התיבה הנפתחת API Type, למרות שבאחריותך לדעת אילו קבועים שייכים לפונקציות הנבחרות.

לאחר שסיימת לעיין ברשימת הפונקציות, לחץ על לחצן Copy. פעולה זו ממקמת את כל הפריטים שנבחרו בלוח. אחר, התחל פרויקט Visual Basic חדש מסוג Standard EXE והוסף מודול חדש. פתח את חלון הקוד של המודול החדש, ולחץ על Paste מתוך תפריט Edit. ההכרזה לפונקציית GetDiskFreeSpace תופיע בחלון הקוד:

```
Private Declare Function GetDiskFreeSpace Lib "kernel32" Alias "GetDiskFreeSpaceA" _
        (ByVal lpRootPathName As String, lpSectorsPerCluster As Long, _
        lpBytesPerSector As Long, lpNumberOfFreeClusters As Long, _
        lpTtoalNumberOfClusters As Long) As Long
```

עכשיו שהתוכנית כוללת את ההכרזה, ניתן לקרוא לפונקציית GetDiskFreeSpace מתוך קוד התוכנית. הוסף לחצן פקודה לטופס, והוסף קוד לשגרת אירוע Click של לחצן זה כדי לקרוא לפונקציה GetDiskFreeSpace, כפי שמוצג בדוגמה הבאה:

```
Private Sub Command1_Click()
        Dim lSecPerClust As Long
        Dim lBytesPerSec As Long
        Dim lFreeClust As Long
        Dim lTotalClust As Long
        Dim lReturn As Long

        lReturn = GetDiskFreeSpace("C:\", lSecPerClust, lBytesPerSec, lFreeClust, _
                lTotalClust)

        Msgbox "Free Clusters on drive C = " & lFreeClust

End Sub
```

הדוגמה הקודמת קוראת לפונקציה GetDiskFreeSpace עם הפרמטרים המתאימים, ומציגה את אחד הערכים המוחזרים בתיבת הודעה. עם זאת דוגמה זו אינה שימושית במיוחד, סביר להניח כי תהיה מעוניין להציג את מספר הבתים החופשיים במקום את מספר האשכולות (Clusters). כמו כן, אם אתה מתכוון להשתמש בפונקציה זו מספר פעמים בתוכניתך, תידרש להקליד קוד רב נוסף כדי לקבל בכל פעם את תוצאות הפונקציה. בסעיף הבא נפתור את שתי הבעיות הללו על ידי יצירת פונקציית "עטיפה".

יצירת פונקציית עטיפה

לעיתים קרובות יוצרים מפתחים פונקציית **עטיפה** (Wrapper) מסביב לקריאת API אחת או יותר. פונקציית עטיפה היא פונקציית Visual Basic רגילה, המטפלת בקריאות שמבצעת התוכנית שלך לפונקציות API. למרות שהשימוש בפונקציה מסוג זה נשמע כצעד מיותר, אחד מהשימושים של פונקציה זו הוא לבטל את הצורך בפעולות נוספות הקשורות בקריאה לפונקציות API. לדוגמה, נניח שאתה כותב תוכנית התקנה ומעוניין לבדוק את גודל שטח האחסון הפנוי על גבי דיסק קשיח מסוים. פונקציית API GetDiskFreeSpace מחזירה מידע אודות השטח הפנוי בדיסק, אך לא במבנה לו הינך זקוק. פונקציה פשוטה המוגדרת על ידי המפתח בשם dblFreeBytes, המוצגת בתוכנית 20.1, קוראת לאותה פונקציית API המשמשת לבדיקת המקום הפנוי בדיסק, אך מחזירה רק את המידע הנחוץ.

תוכנית 20.1: APIWRAP.VBP - פונקציית העטיפה מחזירה רק מידע נחוץ לתוכנית.

```
Declare Function GetDiskFreeSpace Lib "kernel32" Alias "GetDiskFreeSpaceA" _
            (ByVal lpRootPathName As String, lpSectorsPerCluster As Long, _
            lpBytesPerSector As Long, lpNumberOfFreeClusters As Long, _
            lpTtoalNumberOfClusters As Long) As Long

Function dblFreeBytes(sPath As String) As Double
      Dim sDrive As String
      Dim lReturn As Long
      Dim l1 As Long    'l1 = Sectors Per Clusters
      Dim l2 As Long    'l2 = Bytes Per Sector
      Dim l3 As Long    'l3 = Number Of Free Clusters
      Dim l4 As Long    'l4 = Total Number Of Clusters

      sDrive = Left$(sPath, 1) & ":\"  'Get drive letter from path

      lReturn = GetDiskFreeSpace(sDrive, l1, l2, l3, l4)

      dblFreeBytes = l1 * l2 * l3

End Function
```

כפי שניתן לראות מקטע הקוד הקודם, הפונקציה GetDiskFreeSpace מחזירה ארבעה ערכים. אם היית קורא לפונקציה זו מספר פעמים בתוכניתך, היית נאלץ להוסיף ארבע פקודות Dim לכל אחת מהקריאות, ולדעת מהו תפקיד כל אחד מארבעת הפרמטרים שמקבלת הפונקציה. לעומת זאת, השימוש בפונקציה dblFreeBytes, אשר מבצעת את כל הפעולות הדרושות באופן אוטומטי, הוא פשוט הרבה יותר:

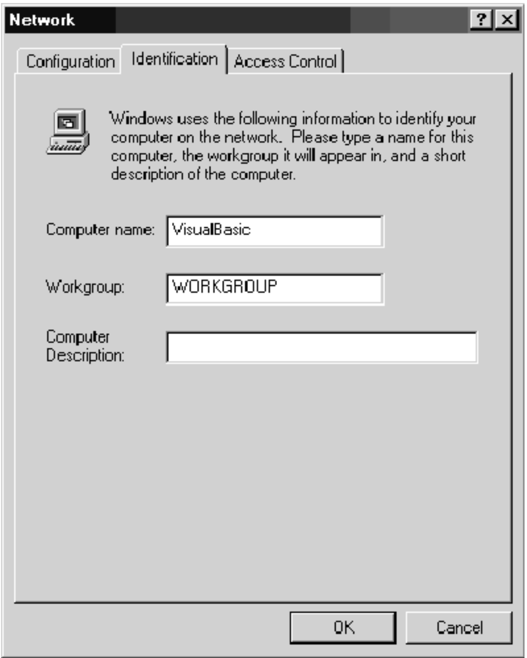
```
If dblFreeBytes(App.Path) < REQUIRED_BYTES Then
      MsgBox "Error: Not Enough Disk Space!"
End IF
```

פונקציות עטיפה יכולות לשמש גם כדי לחסוך פעולות עיבוד מחרוזות הנדרשות על ידי פונקציות API רבות. פונקציות API המחזירות מחרוזות למשתמש, דורשות שיספקו להן כפרמטר את אורך המחרוזת וכן שהמחרוזת עצמה המועברת לפונקציה תמולא בתווי רווח (Space). דוגמה לשימוש בפונקציית עטיפה אשר מבצעת פעולות אלו ניתן לראות בפרק 21 **עבודה עם קבצים**.

ראה "השימוש בקבצי INI ב- Visual Basic" בפרק 21.

יצירת מחלקת עטיפה

בהתאם לרוח תכנות מונחה-עצמים, ניתן לקחת את רעיון פונקציית העטיפה צעד אחד קדימה וליצור מחלקת עטיפה. אתה יכול להשתמש במחלקת הדוגמה ComputerInfo, כדי לשנות את שם המחשב כפי שהוא מוגדר במערכת ההפעלה. ניתן לשנות את שם המחשב בצורה ידנית על ידי שימוש בלוח הבקרה, ביישומון Network (רשת), כפי שמוצג בתרשים 20.3.



תרשים 20.3: קריאות API במחלקת ComputerInfo יכולות לקבוע את שם המחשב

שתי פונקציות API משמשות לאחזור וקביעת שם המחשב: GetComputerName ו-SetComputerName. המחלקה שתיצור תעטוף את שתי הפונקציות הללו, ותהפוך אותן למאפיין אחד אשר בו יהיה קל יותר להשתמש מתוך Visual Basic. תוכנית 20.2 מציגה את הקוד למחלקה ComputerInfo, אשר כוללת מאפיין אחד ואירוע אחד.

תוכנית 20.2: COMPNAME.VBP - מחלקת ComputerInfo קובעת את שם המחשב תחת Windows, הקוד צריך להימצא בתוך מודול מחלקתי - Class Module.

```
Option Explicit
'Notice these API declarations are private - we don't want them accessed directly
Private Declare Function GetComputerName Lib "kerne132" Alias "GetComputerNameA" _
            (ByVal IpBuffer As String, nSize As Long) As Long
Private Declare Function SetComputerName Lib "kerne132" Alias "SetCompterNameA" _
            (ByVal IpComputerName As String) As Long

Public Event RebootNeeded()

Public Property Get ComputerName() As String
      Dim sName As String
      Dim lRetVal As Long
      Dim iPos As Integer

      'API's generally require fixed-length strings, which means
      'pre-filling them with spaces, nulls, or
      Dim sName As String * 255

      'Actually call the API
      lRetVal = GetComputerName(sName, 255&)

      'If the API returns 0 then it didn't work, so exit
      If IRetVal = 0 Then Exit Property

      'GetComputerName puts a null character, Chr$(0), on the end of the
      'string, so we need to remove it before returning the computer name.
      iPos = InStr(sName, Chr$(O))
      ComputerName = Left$(sName, iPos - 1)

End Property

Public Property Let ComputerName(ByVal sNewName As String)

      Dim lRetVal As Long
      lRetVal = SetComputerName(sNewName)
      RaiseEvent RebootNeeded

End Property
```

בדוגמה שהובאה לעיל, הסיבה ליצירת אירוע RebootNeeded היא כפולה: ראשית, למרות שניתן לאמת את השם החדש של המחשב על ידי בדיקה בלוח הבקרה, פונקציית GetCompterName תמשיך להחזיר את השם הקודם עד לביצוע אתחול.

שנית, אם המחשב מחובר לרשת, יש צורך לאתחל את המחשב כדי לרשום את שמו החדש ברשת. בכל מקרה, נעשה שימוש באירוע מותאם אישית, כדי שהתוכנית שמבצעת את הקריאה תבצע את הפעולות הנדרשות. דוגמה לקטע קוד שנועד לבחון את פעולתה של מחלקת CompterInfo, מוצגת בתוכנית 20.3.

תוכנית 20.3: COMPNAME.VBP - בדיקת מחלקת CompterInfo מתוך טופס.

```
Dim WithEvents clsCompName As ComputerInfo

Sub TestComputerInfoClass()

    Dim sName As String

    'Create a new instance of the class
    Set clsCompName = New ComputerInfo

    'Display the computer name
    sName = clsCompName.ComputerName
    MsgBox "The current computer name is: " & sName

    'Set a new Computer name
    sName = InputBox$("Enter new computer name")
    clsCompName.ComputerName = sName

End Sub

Private Sub clsCompName_RebootNeeded()
    MsgBox "You need to reboot to make the changes effective!"
End Sub
```

מחלקת ComuterInfo היא דוגמה טובה לשימוש ביכולות של Visual Basic כשפת תכנות מונחה-עצמים. בנוסף על כך שמחלקה זו הופכת את השימוש בפונקציית ComputerName לפשוטה יותר, ניתן להשתמש בה במספר פרויקטים לאחר שנוצרה בפעם הראשונה. כן, ניתן להוסיף מאפיינים ושגרות אירוע למחלקה זו על פי הצורך.

קריאות API שימושיות

הדרך הטובה ביותר ללמוד אודות API של Windows היא להתנסות בשימוש בו. אם תסתכל בקובץ WIN32API.TXT תראה שקיים מספר גדול למדי של פונקציות API. הסעיפים הבאים יציגו בפניך את הפונקציות המעניינות ביותר.

קריאות API "מהנות"

שתי פונקציות API הבאות שתכיר, הן פשוטות לשימוש ומהנות. הפונקציה הראשונה, sndPlaySound, מאפשרת לנגן קובץ קול בתבנית WAV. באמצעות כרטיס קול. פונקציה זו מקבלת שני פרמטרים: שם הקובץ, ודגלים (Flags) הקובעים את הדרך בה מנוגן הקובץ.

תוכנית 20.4: WAVDEMO.VBP **ניגון קובץ** WAV.

```
Private Declare Function sndPlaySound Lib "winmm.dll" Alias "sndPlaySoundA" (ByVal _
                    lpszSoundName As String, ByVal uFlags As Long) As Long

Public Const SND_ALIAS = &H10000         ' name is in WIN.INI or the Registry
Public Const SND_ASYNC = &H1             ' play asynchronously
Public Const SND_SYNC = &H0              ' play synchronously (default)
Public Const SND_NOWAIT = &H2000         ' don't wait if the driver is busy
Public Const SND_LOOP = &H8              ' loop the sound until next sndPlaySound

Sub SoundCheck()
    Dim lRetVal As Long

    lRetVal = sndPlaySound("C:\WINDOWS\MEDIA\CHIMES.WAV", SND_SYNC)
    lRetVal = sndPlaySound("SystemStart", SND_ALIAS + SND_ASYNC + _
        SND_NOWAIT)

    'The alias names of system sounds are listed in the registry together,
    'to find them search for SystemStart

End Sub
```

כל הקבועים המתחילים בקידומת SND רשומים בקובץ הטקסט של API בצירוף הערות. שני הקבועים החשובים הם SND_ASYNC ו-SND_SYNC, אשר קובעים האם התוכנית תמתין עד לסיום נגינת הקובץ לפני שתמשיך לבצע את שאר הקוד.

פונקציית API אחרת בשם SystemParametersInfo, יכולה לשמש כדי לקבוע את דוגמת הרקע של Windows. השתמשתי בפונקציה זו ובמצלמת ללכידת וידאו כדי ליצור "חלון משרד וירטואלי" במקום עבודתי. דוגמה המשתמשת בפונקציה SystemParametersInfo מוצגת בתוכנית 20.5.

תוכנית 20.5: BKGDEMO.VBP - פונקציית SystemParametersInfo יכולה לקבוע את הרקע של שולחן העבודה של Windows.

```
Public Const SPIF_UPDATEINIFILE = &H1
Public Const SPIF_SENDWININICHANGE = &H2
Public Const SPIF_SETDESKWALLPAPER = 20
Declare Function SystemParametersInfo Lib "user32" Alias "SystemParametersInfoA" _
                    (ByVal uAction As Long, ByVal uParam As Long, ByVal lpbParam _
                     As Any, ByVal fuWinIni As Long) As Long

Sub ClearWallpaper()

    Dim lRetVal As Long
    lRetVal = SystemParametersInfo(SPIF_SETDESKWALLPAPER, 0&, "(None)", _
            SPIF_UPFATEINIFILE Or SPIF_SENDWININICHANGE)

End Sub

Sub SetWallPaper(sBitmapFile As String)

    Dim lRetVal As Long
    lRetVal = SystemParametersInfo(SPIF_SETDESKWALLPAPER, 0&, sBitmapFile, _
                SPIF_UPFATEINIFILE Or SPIF_SENDWININICHANGE)

End Sub
```

הפונקציות SetWallpaper ו-ClearWallpaper גורמות למפת הסיביות המשמשת כרקע שולחן העבודה של Windows להשתנות באופן מיידי. הקבוע המועבר כפרמטר האחרון לפונקציה, למרות שאינו הכרחי לפעולתה, גורם לשינויים הנעשים, להופיע גם בלוח הבקרה. אם תמשיך לעיין בקובץ הטקסט של API, תוכל לראות שניתן לשלוט במאפייני תצוגה נוספים על ידי שימוש בפונקציה SystemParametersInfo.

איתור ושליטה על חלונות אחרים

ניתן לשלוט בצורת התנהגות חלון, ולקבל מידע על חלונות אחרים באמצעות מספר קריאות API. רוב קריאות API אלו משתמשות **במזהה ייחודי של חלון** (Window Handle) כדי לזהות את החלון איתו אתה מעוניין לעבוד. מזהה ייחודי של חלון הוא מספר ייחודי המוקצה לכל חלון פתוח במערכת. בתוכניות Visual Basic ניתן לקבל את המזהה הייחודי של חלון טופס כלשהו, מתוך מאפיין hWnd של הטופס. עם זאת, אם תרצה לפנות לחלונות מחוץ לתוכנית Visual Basic שלך, תהיה חייב להשתמש בקריאת API אשר תחזיר את המזהה הייחודי של החלון המבוקש.

הערה:

בסעיף זה נחרוג מהרגלנו להוסיף את האות l קידומת למשתנים ארוכים (long). המזהה הייחודי של חלון הוא תמיד מסוג long, והכרזות API מתייחסות אליו בדרך כלל בשם hWnd (קיצור Handle to Window).

איתור מזהה ייחודי של חלון

כדי למצוא את המזהה הייחודי לחלון מסוים ניתן להשתמש בפונקציית API FindWindow. פונקציה זו מקבלת שני פרמטרים: שם החלון (Window Name) ואת שם מחלקת החלון (Window Class Name). אפשר להעביר לפונקציה זו אחד משני פרמטרים אלה, או את שניהם, והפונקציה תחזיר את המזהה הייחודי של החלון, או אפס אם לא קיים חלון מתאים.

לדוגמה, הפעל את היישום Notepad (פנקס הרשימות) ותוכל לראות כי הכותרת של חלון התוכנית היא Untitled – NotePad. כדי למצוא את המזהה הייחודי לחלון זה ניתן לקרוא לפונקציית FindWindow בצורה הבאה:

```
hwndNotepad = FindWindow(vbNullString, "Untitled – NotePad")
```

שים לב כי בדוגמה זו אנו מעבירים לפונקציה רק את שם החלון, ומעבירים מחרוזת ריקה כערך לפרמטר שם מחלקת החלון. אולם נניח כי שמרת או פתחת קובץ מתוך NotePad. כותרת החלון של NotePad תשתנה כדי להציג את שם הקובץ החדש, במקרה כזה, שורת הקוד הקודמת לא תחזיר את המזהה הייחודי לחלון NotePad שכן כעת כבר לא קיים חלון פתוח שכותרתו היא "Untitled – NotePad". עם זאת, במקרה אנו יודעים כי שם המחלקה של חלון Notepad הוא Notepad, לכן אפשר לקרוא לפונקציה בצורה שונה:

```
hwndNotepad = FindWindow("Notepad", vbNullString)
```

טיפ:

שמות המחלקות של Windows אינם תמיד כה ברורים. לדוגמה, שמה של אחת ממחלקות הטפסים המוקדמות של Visual Basic היה ThunderForm. עם זאת, תוכנת "ריגול" אחר מחלקות Windows, כדוגמת התוכנה המצורפת אל Visual C++ מסייעת רבות בתהליך מציאת השמות.

שורת הקוד הקודמת מסוגלת למצוא את המזהה הייחודי של החלון NotePad ללא קשר עם כותרת החלון. דוגמה זו עובדת כיאות במקרה שקיים רק חלון NotePad אחד, אך מה קורה אם קיימים מספר עותקים של חלון זה בו זמנית? התשובה לכך טמונה במספר קריאות API נוספות: GetWindow (קבל חלון), GetWindowText (קבל טקסט חלון), וכן GetWindowClass (קבל מחלקת חלון).

בדומה לפונקציה FindWindow, משמשת הפונקציה GetWindow לאיתור המזהים הייחודיים לחלונות. אך במקום לאתר את החלון המבוקש על פי שמו, מוצאת GetWindow את החלון בהתבסס על יחסו לחלונות אחרים. ניתן להשתמש בפונקציה GetWindow כחלק מלולאה, כדי לעיין ברשימת החלונות הקיימים כפי שמוצג בתוכנית 20.6. הקוד בתוכנית זו סופר את מספר המופעים של חלון NotePad, ומדפיס כותרת של כל חלון.

תוכנית 20.6: WINFIND.VBP - איתור כל החלונות של NotePad.

```
Public Const GW_HWNDFIRST = 0
Public Const GW_HWNDLAST = 1
Public Const GW_HWNDNEXT = 2
Public Const GW_HWNDPREV = 3
Public Const GW_OWNER = 4
Public Const GW_CHILD = 5
Public Const GW_MAX = 5

Declare Function GetWindow Lib "user32" (ByVal hwnd As Long, ByVal wCmd As Long) _
        As Long
Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal lpClassName _
        As String, ByVal lpWindowName As String) As Long
Declare Function GetClassName Lib "user32" Alias "GetClassNameA" (ByVal hwnd _
        As Long, ByVal lpClassName As String, ByVal nMaxCount As Long) As Long
Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" (ByVal hwnd _
        As Long, ByVal lpString As Long, ByVal cch As Long) As Long

Sub TestFindWindow()

    Const MAX_TEXT_SIZE = 50

    Dim hWndCurrent As Long
    Dim iCount As Integer
    Dim sClass As String
    Dim sCaption As String
    Dim lRetVal As Long

    iCount = 0
    hWndCurrent = FindWindow("Notepad", vbNullString)

    Do While hWndCurrent <> 0

        'Get the class name of the current Window
        sClass = Space$(MAX_TEXT_SIZE + 1)
        lRetVal = GetClassName(hWndCurrent, sClass, MAX_TEXT_SIZE)

        'If the Window is Notepad then increment count, print caption
        If Trim$(sClass) = "Notepad" & Chr$(0) Then
            iCount = iCount + 1
            sCaption = Space$(MAX_TEXT_SIZE + 1)
            lRetVal = GetWindowText(hWndCurrent, sCaption, MAX_TEXT_SIZE)
            Debug.Print Left$(sCaption, lRetVal)
        End If

        hWndCurrent = GetWindow(hWndCurrent, GW_HWNDNEXT)

    Loop
    Debug.Print "Number of open Notepad windows: " & iCount
End Sub
```

אזהרה:

יש להבין את תפקיד תו Null בקריאות API. בתוכנית 20.6, שתי הפונקציות GetClassName ו-GetWindowText, ממקמות ערך Null בסוף המחרוזת המוחזרת. מסיבה זו הוספנו Null בסוף המילה Notepad במשפט If, כך שהשוואת המחרוזות תתבצע כראוי. פונקציית Trim (קצוץ) מסירה תווי רווח, אך לא את תו Null המסיים את המחרוזת. דרך נוספת לפתור בעיה זו היא להשתמש בערך שמחזירה הפונקציה GetClassName עם פונקציית Left$ כפי שמודגם בשורת הקוד:

```
If Left$(sClass, lRetVal) = "Notepad" Then ...
```

בכל אופן, קרוב לוודאי שבעיה זו תחייב אותך לכתוב פונקציה מהירה, להסרת תווי רווח ותווי Null מיותרים ממחרוזות.

השימוש במזהה הייחודי של החלון

אם כן עכשיו, לאחר שהצלחנו לבסוף להשיג את המזהה הייחודי של החלון, מה בעצם ניתן לעשות איתו? התשובה היא, די הרבה. ראשית ניתן לשלוט במצבו של החלון באמצעות שימוש בפונקציית ShowWindow API :

```
lRetVal = ShowWindow(hwndNotepad, SW_SHOWMINIMIZED)
```

ניתן גם להשתמש בקריאות SendMessage API ו-PostMessage, כדי לשלוח הודעות Windows לחלון מסוים. לדוגמה, שורת הקוד הבאה תגרום לחלון Notepad להיסגר כאילו סיימת את התוכנית מתוך מנהל המשימות של Windows :

```
lRetVal = SendMessage(hwndNotepad, WM_CLOSE, 0&, 0&)
```

שורת קוד זו גורמת למערכת ההפעלה לשלוח הודעת WM_CLOSE לחלון שצוין כפרמטר הראשון לפונקציה. ברגע קבלת ההודעה, Notepad תסיים פעולתה, ייתכן שלאחר שתשאל את המשתמש באם רצונו לשמור מסמך פתוח. תחביר קריאת PostMessage שווה לזה של SendMessage, וההבדל ביניהן הוא שהפונקציה PostMessage פועלת בצורה א-סינכרונית, כלומר היא אינה ממתינה עד שההודעה שנשלחה תעובד על ידי החלון המקבל לפני שתמשיך בביצוע התוכנית.

שים לב כי בדוגמה שהוצגה לעיל, שני הפרמטרים האחרונים הם אפס, שכן פרמטרים אלה הם ייחודיים להודעות מסוימות, ואילו הודעת WM_CLOSE אינה משתמשת בהם. לדוגמה, אם תשתמש בהודעת WM_CHAR כדי לשלוח תו לחלון מסוים, ישמשו שני פרמטרים אלה לקביעת התו אותו יש לשלוח. זכור כי ניתן לשלוח מאות הודעות שונות.

אזהרה:

למרות שהדוגמאות המוצגות כאן בטוחות יחסית כאשר משתמשים בהן עם תוכנת Notepad, תוכניות אחרות עלולות להגיב בצורה בלתי רצויה ולגרום לשגיאות. לקבלת התוצאות הטובות ביותר במהלך השימוש בקריאות API עם חלונות אחרים:

1. קרא והבן את תיעוד פונקציית API ו/או הודעת Windows בהן אתה משתמש,
2. ודא כי אתה עובד עם החלון הנכון,
3. הכן עצמך לביצוע אתחולים רבים,
4. דאג לשמור את תוכניתך לפני הרצת בדיקה.

המתנה לסיום פעולת תוכנית

לעיתים ייתכן כי תצטרך להפעיל תוכניות אחרות מתוך Visual Basic. לצורך כך קיימת פונקציה מובנית ב- Visual Basic בשם **Shell**, השימוש בפונקציה זו מתואר בפרק 21, **עבודה עם קבצים**.

ראה "שימוש בפונקציה Shell להפעלת תוכניות נוספות" בפרק 21.

עם זאת, החיסרון הקיים בשימוש בפונקציה Shell הוא שהיא אינה ממתינה לסיום התוכנית שהופעלה, לפני ביצוע שורת הקוד הבאה:

```
'THIS DOES NOT WORK!!!
dRetVal = Shell("MyBat.Bat")
MsgBox "Batch file complete!"
```

קטע קוד זה אינו מבצע את המצופה ממנו. תיבת ההודעה מופיעה ברגע שפקודת Shell מבוצעת, בין אם ביצועו של קובץ האצווה הסתיים או לא. בעולם 16Bit של Windows 3.11 ו- Visual Basic 3.0, היה אפשר לעקוף בעיה זו בקלות בעזרת מספר שורות קוד ושתי קריאות API. אך בעולם 32Bit, הדברים מסובכים הרבה יותר. כיסוי מעמיק של תהליכים והליכי משנה במערכת 32Bit, ידרוש בוודאי חצי ספר. במקום להיכנס לכל הפרטים האלה, נציג בקצרה פונקציה אשר תבצע משימה זו, ראה תוכנית 20.7.

תוכנית 20.7: SHELLWAIT.VBP המתנה לסיום פעולת תוכנית לפי שיטת Microsoft.

```
Private Type STARTUPINFO
        cb As Long
        lpReserved As String
        lpDesktop As String
        IpTitle As String
        dwX As Long
```

```
        dwY As Long
        dwXSize As Long
        dwYSize As Long
        dwXCountChars As Long
        dwYCountChars As Long
        dwFillAttribute As Long
        dwFlags As Long
        wShowWindow As Integer
        cbReserved2 As Integer
        pReserved2 As Long
        hStdInput As Long
        hStd0utput As Long
        hStdError As Long
End Type
Private Type PROCESS_INFORMATION
        hProcess As Long
        hThread As Long
        dwProcessID As Long
        dwThreadID As Long
End Type

Private Declare Function WaitForSingle0bject Lib "kernel32" (ByVal _
  hHandle As Long, ByVal dwMilliseconds As Long) As Long
Private Declare Function CreateProcessA Lib "kerne132" (ByVal IpApplicationName As _
                Long, ByVal IpCommandLine As String, ByVal IpProcessAttributes As Long, _
                ByVal IpThreadAttributes As Long, ByVal bInheritHandles As Long, ByVal _
                dwCreationFlags As Long, ByVal IpEnvironment As Long, ByVal _
                IpCurrentDirectory As Long, IpStartupInfo As STARTUPINFO, _
                IpProcessInformation As PROCESS_INFORMATION) As Long

Private Declare Function CloseHandle Lib "kerne132" (ByVal  hObject As Long) As Long
Private Const NORMAL PRIORITY_CLASS = &H20&
Private Const INFINITE = -1&
Public Sub ExecCmd(cmdline$)
        Dim proc As PROCESS_INFORMATION
        Dim Start As STARTUPINFO

        'Initialize the STARTUPINFO structure:
        Start.cb = Len(Start)

        'Start the shelled application:
        ret& = CreateProcessA(0&, cmdline$, 0&, 0&, 1&, NORMALPRIORITY_CLASS, _
                0&, 0&, Start, proc)
        'Wait for the shelled application to finish:
        ret& = WaitForSingleObject(proc. hProcess, INFINITE)
        ret& = CloseHandle(proc.hProcess)
End Sub
```

באמצעות שימוש בפונקציית ExecCmd מתוכנית 20.7, ניתן לשכתב את הדוגמה מתחילת סעיף זה בצורה הבאה:

```
'THIS WORKS AS INTENDED
ExecCmd "MyBat.Bat"
MsgBox "Batch file complete!"
```

שים לב כי ביצוע משימה זו דורש שימוש במספר לא קטן של הכרזות API, אך פונקציית ExecCmd מפשטת תהליך זה.

קריאות חוזרות ומחלקות משנה

קריאות API מאפשרות לך לעבוד עם מערכת ההפעלה Windows ברמה נמוכה יותר ממה שמאפשר קוד Visual Basic סטנדרטי. בכתיבת תוכניות ב- Visual Basic, אתה בעצם משתמש בקוד Visual Basic כמעטפת לקריאות API של Windows. למרות שרוב התוכניות שתכתוב לא תדרושנה שימוש בקריאות API מסובכות, לעיתים יעלה הצורך להשתמש בקריאות API, אם המשימה אותה אתה מנסה לבצע לא יושמה ישירות ב- Visual Basic. תחום אחד בו אתה עשוי להזדקק לקריאות API הוא עיבוד הודעות Windows, בהן ניתן לטפל באמצעות **קריאות חוזרות** (Callbacks) **ומחלקות משנה** (Subclassing). טכניקות אלו מתוארות בסעיפים הבאים.

קריאות חוזרות

קריאה חוזרת, היא פונקציה בתוכנית שלך אשר נקראת על ידי Windows. ראה לדוגמה את ההכרזה על פונקציית API בשם EnumWindows:

```
Declare Function EnumWindows Lib "user32" (ByVal lpEnumFunc As Long, LParam as _
            Any) As Long
```

שים לב כי הפרמטר הראשון לפונקציה EnumWindows הוא מצביע לפונקציה. הפונקציה שאליה מצביע פרמטר זה, אמורה להיכתב בתוך מודול Visual Basic, והיא פונקציית הקריאה החוזרת שלך. פונקציית EnumWindows עוברת על רשימת המשימות של Windows, ומבצעת את פונקציית הקריאה החוזרת שלך עבור כל פריט ברשימה זו. כדי להעביר מצביע אל EnumWindows, יש להשתמש במילת המפתח AddressOf ובשם הקריאה החוזרת, כפי שמוצג בדוגמה הבאה:

```
lRetVal = EnumWindows(AddressOf MyCallBackFunc, 12345&)
```

ביצוע שורת קוד זו (פעם אחת בלבד) גורם ל-EnumWindows לקרוא לפונקציה MyCallBackFunc פעם אחר פעם עבור כל חלון ברשימת המשימות. כאשר EnumWindows קוראת לפונקציה MyCallBackFunc, היא מעבירה לה שני פרמטרים: הראשון הוא מזהה ייחודי לחלון, והשני הוא הפרמטר השני שהועבר במקור אל EnumWindows. (הפרמטר השני יכול לשמש את פונקציית הקריאה החוזרת שלך בכל דרך שתמצא לנכון, בדומה לפרמטר ההודעה של פונקציית SendMessage). מסיבה זו קריאות חוזרות נכתבות במיוחד עבור פונקציות API אשר קוראות להן. במילים

אחרות, פונקציית קריאה חוזרת עבור הפונקציה EnumWindows חייבת להיות מוכרזת על פי כללים מסוימים. פונקציה כזו חייבת לקבל שני פרמטרים ולהחזיר ערך מסוג Long. הערך שמחזירה פונקציה זו משמש לעצירת הפונקציה EnumWindows, לפני שהיא מסיימת לספור את כל החלונות הפעילים.

תוכנית 20.8 מציגה גירסה פשוטה של הפונקציה MyCallBackFunc, הניתנת לשימוש עם הפונקציה EnumWindows. במקרה זה, ערך הפרמטר השני מייצג שם טופס. בכל פעם שמתבצעת קריאה לפונקציה MyCallBackFunc, היא מדפיסה נתונים אודות המזהה הייחודי של החלון שהועבר אליה כפרמטר הראשון, על גבי הטופס.

תוכנית 20.8: CALLBACK.VBP - הדגמת פונקציית קריאה חוזרת.

```
Declare Function EnumWindows Lib "user32" (ByVal lpEnumFunc As Long, lParam As _
            Any) As Long

Function MyCallBackFunc(ByVal hwnd As Long, lParam As Form) As Long

      Dim sInfo As String

      'sGetWindowInfi is a wrapper
      'for GetClassName and GetWindowText

      sInfo = sGetWindowInfo(hwnd)

      lParam.Print "hwnd=" & hwnd & vbTab & sInfo

      'Returning False causes EnumWindows to stop
      'moving through the window list
      MyCallBackFunc = True

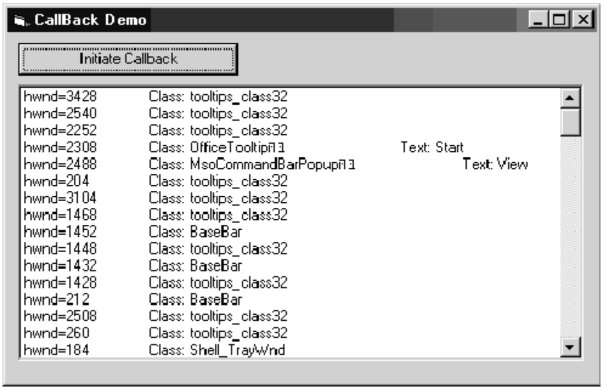
End Function

Sub InitiateCallback()
      'This code is called from the form's command button

      Dim lRetVal As Long
      Form1.Cls
      lRetVal = EnumWindows(AddressOf MyCallBackFunc, Form1)

End Sub
```

ניתן לראות דוגמה לפעולתה של תוכנית זו בתרשים 20.4.



תרשים 20.4: הפונקציה EnumWindows מעבירה לפונקציית הקריאה החוזרת שלך את המזההים הייחודיים של כל החלונות שנמצאים ברשימת המשימות

מחלקות משנה

עתה, לאחר שסקרנו את נושא הפונקציות החוזרות, נראה רעיון דומה הנקרא **מחלקות משנה** (Subclassing). כדי להבין רעיון זה, יש להבין ראשית את עיקרון הטופס הסטנדרטי של Visual Basic. ניתן לחשוב על טופס כעל סוג מיוחד (או מחלקה) של חלון. סוג זה של חלון תוכנת "ליירט" אירועים מסוימים, כגון אירוע MouseMove (תנועת עכבר). טופס Visual Basic "יודע" שעליו להפעיל את שגרת האירוע MouseMove מכיון שהוא תוכנת לעשות כך. אך מה קורה עם כל הודעות Windows שמהן הטופס מתעלם? ומה אם אתה מעונין לשנות את הדרך בה Windows מטפלת בהודעות קיימות?

כדי ליירט ולהגיב לאירועים, עליך לשנות את הדרך בה מחלקת הטופס מטפלת באירועים. ניתן לעשות כן באמצעות טכניקה שנקראת יצירת מחלקות משנה, אשר בגרסאות Visual Basic שקדמו לגירסה 5.0, ניתן היה ליישם רק באמצעות פקדים מותאמים אישית. עם זאת, עתה כשניתן להשתמש בפונקציות קריאה חוזרת, ניתן להשתמש בפונקציית SetWindowLong API עם פונקציית קריאה חוזרת, כדי ליירט הודעות Windows.

יצירת מטפל באירועים (Event Handler) עם מחלקות משנה

ניתן לראות דוגמה טובה לשימוש במחלקות משנה במאמר **העברת מצביעי פונקציות לפרוצדורות DLL וספריות סוגים** (Passing Function Pointers to DLL Procedures and Type Libraries), אשר כלול בקבצי העזרה של Windows. הקוד לדוגמה זו מופיע ברשימות 20.9 ו-20.10, ומייד אחריהן הסברים מפורטים.

הערה:

זכור לשמור את עבודתך לעיתים קרובות כאשר אתה משתמש בפונקציות API.

תוכנית 20.9: SUBCLASS.VBP - דוגמה לשימוש בתת-מחלקות

```
Declare Function CallWindowProc Lib "user32" Alias "CallWindowProcA" _
		(ByVal lpPrevWndFunc As Long, ByVal hwnd As Long, ByVal Msg As Long, _
		ByVal wParam As Long, ByVal lParam As Long) As Long
Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hwnd _
		As Long, ByVal nIndex As Long, ByVal deNewLong As Long) As Long

Global Const GWL_WNDPROC = -4
Public lpPrevWndProc As Long
Public gHW As Long

Function WindowProc(ByVal hw As Long, ByVal uMsg As Long, ByVal wParam As Long,_
		ByVal lParam As Long) As Long
	Debug.Print "Message: "; hw, uMsg, wParam, lParam
	WindowProc = CallWindowProc(lpPrevWndProc, hw, uMsg, wParam, lParam)
End Function

Public Sub Hook()
	lpPrevWndProc = SetWindowLong(gHW, GWL_WNDPROC, AddressOf _
		WindowProc)
End Sub

Public Sub Unhook()
	Dim temp As Long
	temp = SetWindowLong(gHW, GWL_WNDPROC, lpPrevWndProc)
End Sub
```

למרות שהקוד בתוכנית 20.9 משמש ליירוט הודעות הנשלחות לטופס, קיים חוק כללי לפונקציות קריאה חוזרת שמכתיב כי מיקום פונקציות אלו הוא בתוך מודול, ולכן יש להקליד קטע קוד זה כחלק ממודול ולא כקוד של הטופס. חלקו השני של הקוד הנדרש ליישום דוגמה זו, אשר אותו יש להקליד כקוד של הטופס, מוצג בתוכנית 20.10.

תוכנית 20.10: SUBCLASS.VBP - קוד רמת-הטופס לדוגמת מחלקות המשנה

```
Private Sub cmdHook_Click()
	Hook
End Sub

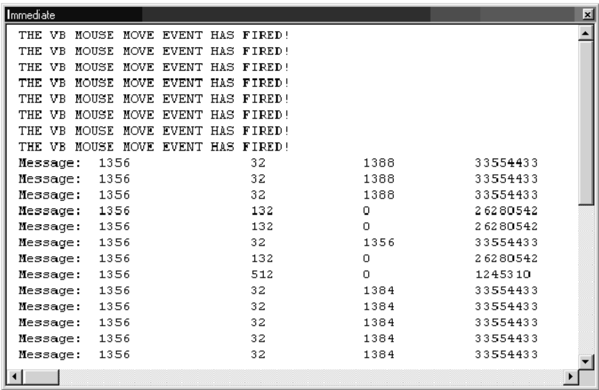
Private Sub cmdUnhook_Click()
	Unhook
End Sub

Private Sub Form_Load()
	gHW = Me.hwnd
End Sub
```

כדי לעבור על פעילות הקוד המוצג ברשימות אלו צעד-אחר-צעד, נתחיל עם תת-השיגרה Hook בתוכנית 20.9. Hook מעבירה את כתובתה של פונקציית הקריאה החוזרת WindowProc לקריאת SetWindowLong API. פונקציית SetWindowLong מגדירה כי WindowProc היא מטפלת ההודעות (Message Handler) של הטופס, ובכך מחליפה את מטפלת ההודעות של Visual Basic. הכתובת של שגרת הטיפול בהודעות שהוגדרה על ידי Visual Basic, מוחזרת על ידי הפונקציה ומאוחסנת במשתנה lpPrevWndProc. תת-שגרת Unhook קוראת לפונקציה SetWindowLong ומעבירה לה כפרמטר את הערך המאוחסן במשתנה lpPrevWndProc כדי להחזיר את השליטה על אירועים המגיעים לטופס לידי שגרת הטיפול המקורית. תת-שגרות Hook ו-Unhook פועלות כמעין מתג אשר קובע לאיזו שיגרת טיפול בהודעות תישלחנה כל הודעות Windows המיועדות לטופס. לאחר ביצוע קריאה לשיגרה Hook, מנותבות כל הודעות Windows לפונקציה WindowProc.

WindowProc מקבלת ארבעה פרמטרים: המזהה הייחודי של חלון הטופס, ושלושת הפרמטרים המשויכים לכל הודעה. שורת הקוד הראשונה של שגרת WindowProc פשוט מדפיסה נתונים אלה בחלון Immediate. השורה השנייה קוראת לפונקציית API בשם CallWindowProc. פונקציה זו מעבירה את נתוני ההודעה לשגרת הטיפול בהודעות הסטנדרטית, אשר כתובתה מאוחסנת במשתנה lpPrevWndProc.

אם כך, מבחינה מעשית, שגרת טיפול ההודעות שבנינו לא עושה דבר, מלבד להדפיס את ההודעה ולקרוא לשגרת הטיפול הסטנדרטית, אשר היתה נקראת לטפל בהודעה בכל מקרה. דוגמה לפלט תוכנית זו ניתן לראות בתרשים 20.5.



תרשים 20.5: תוכנית לדוגמה, המשתמשת בטכניקות מחלקות משנה כדי להתקין את שגרת הטיפול בהודעות שלה

כדי לוודא שהתוכנית שכתבנו באמת מיירטת הודעות, עלינו לגרום לפונקציה WindowProc לבצע פעולה מיוחדת, כמו למשל להדפיס הודעה. חזור לדוגמת אירוע MouseMove. הקלד את שורת הקוד הבאה בשגרת האירוע MouseMove של הטופס:

```
Debug.Print "THE VB MOUSE MOVE EVENT HAS FIRED!"
```

הפעל את תוכנית הדוגמה שנית, ושים לב כי חלון Immediate מציג את הפלט של פקודת Print זו וכן את הפלט של פקודת Print הנמצאת בשיגרה WindowProc. עם זאת, על ידי שינוי הפונקציה WindowProc, ניתן "לכבות" את אירוע MouseMove של Visual Basic:

```
Function WindowProc(ByVal hw As Long, ByVal uMsg As Long, ByVal wParam As Long,_
            ByVal lParam As Long) As Long
        Debug.Print "Message: "; hw, uMsg, wParam, lParam
        If uMsg <> 512 Then WindowProc = CallWindowProc(lpPrevWndProc, hw, _
            Msg, wParam, lParam)

        '(Decimal 512 is 200 hex which is WM_MOUSEMOVE in the API text file.)
End Function
```

מכאן...

פרק זה בוודאי אינו מהווה מדריך שלם לנושא API. מטרת הפרק היא להציג בפניך מספר נושאים הקשורים לשימוש בפונקציות API של Windows. זכור כי API של Windows אינו ממשק פיתוח היישומים היחיד הקיים. ניתן לרכוש ספריות DLL מיצרן צד שלישי המבצעות מיגוון רחב של פונקציות, כגון יצירת קבצי ZIP (דחיסת קבצים) או יצירת דוחות מותאמים לצורכי המשתמש וכדומה. במקרה כזה, התיעוד והצהרות Declare המתאימות יהיו כלולים עם המוצר.

התיעוד של Windows API ב- Visual Basic לוקה בחסר. כדי להשתמש בפונקציות API אלו בצורה יעילה, קרוב לוודאי שתהיה חייב למצוא מקור אחר למידע רלוונטי. המדריך הסטנדרטי ל-API למפתחי Visual Basic הוא ספרו של דניאל אפלמן Daniel Appleman - **Visual Basic Programmer's Guide to the Win32 API**, המכסה נושא זה בצורה מעמיקה ויסודית. מקור נוסף הוא Microsoft Developer Network Online, אליו ניתן להגיע באתר: **http://msdn.microsoft.com**.

לפרטים נוספים אודות המידע הנמצא בפרק זה, עיין בפרקים הבאים:

- כדי ללמוד אודות קריאות API הקשורות לגרפיקה, עיין בסעיף **שיטות גרפיקה אחרות** בפרק 19 **שימוש במרכיבי התכנון הוויזואלי**.
- לשימוש בקריאות API כדי לגשת לקבצי INI, ראה פרק 21 **עבודה עם קבצים**.
- כדי לראות כיצד קריאות API יכולות לשנות את הדרך בה מוצגות מפות סיביות על המסך, עיין בפרק 13 **עבודה עם מערכי פקדים**.

21

עבודה עם קבצים

מה בפרק זה?

- פונקציות לטיפול בקבצים ב- Visual Basic
- עבודה עם קבצי טקסט
- קבצים אקראיים - יצירת מבני קבצים שלך
- קבצי INI

ללא תלות בשפת התכנות בה תשתמש אתה עשוי להידרש במוקדם או במאוחר לכתוב קוד שיינהל מגעים עם קבצי מערכת ההפעלה. לדוגמה, ייתכן שתידרש להעתיק קבצים או להפעיל תוכנית אחרת. פעולות כאלו תוכל לבצע באופן אינטראקטיבי על ידי שימוש בסייר Windows או בחלון שורת הפקודה של DOS, אך במהלך הפרק גם תיווכח שתוכל לבצע פעולות כאלו בקלות, על ידי שימוש בקוד Visual Basic.

שימוש נוסף בקבצים הוא אחסון ואחזור מידע. בפרקים אחרים למדת כיצד להשתמש במסדי נתונים לאחסון מידע. אך במקרים מסוימים מנגנון ניהול מסד הנתונים של Visual Basic עלול להיות מורכב מדי לצרכיך, או שאופן הפעולה שלו עלול שלא להתאים. לדוגמה, ייתכן שתרצה לנהל קובץ יומן (Log File) פשוט, או לעבד קובץ טקסט פשוט מופרד באמצעות פסיק (Comma-Delimited). בפרק זה נבחן כמה דרכים שתאפשרנה לך לאחסן נתונים תוך שימוש בקבצי טקסט משלושה סוגים: **קבצים סדרתיים** (Sequential Files), **קבצים לגישה אקראית** (Random Access Files) **וקבצי אתחול** (INI - Initialization Files).

פונקציות טיפול בקבצים ב- Visual Basic

קבצים הם יחידות הארגון הבסיסיות של מערכת ההפעלה. קיימים סוגים רבים של קבצים, החל מקבצים שנוצרו על ידי המשתמש, משחקים וכלה בפקודות מערכת ההפעלה. בסעיף זה נדון בחלק מהפונקציות שמשמשות לעבודה עם קבצים במסגרת תוכניות הכתובות ב- Visual Basic. בפרק 30 **שימוש ב-VBScript** נדון בשיטה נוספת לגישה אל קבצים, אשר מיושמת באמצעות ממשק המבוסס על אובייקט FileSystemObject (אובייקט של מערכת הקבצים - סוג אובייקט זה היה בעבר מרכיב של שפת VBScript אך כיום הוא נגיש גם מ- Visual Basic, החל מגירסה 6). הפונקציות שתידונה בסעיף זה הן פונקציות שכלולות ב- Visual Basic זמן כה רב, עד שהן נחשבות לפונקציות "מסורתיות" של השפה.

ראה, "גישה למערכת הקבצים", פרק 30.

פונקציה Dir לאיתור קבצים והצגת רשימות קבצים

הפונקציה $Dir היא פונקציה מועילה בעת עבודה עם קבצים. אופן פעולת הפונקציה דומה לזה של הפקודה Dir שמוקלדת בשורת הפקודה של DOS. הפונקציה $Dir משמשת להצגת רשימת קבצים אשר תואמים להגדרה של שם קובץ או של נתיב (Path). הנתיב המשמש לפנייה אל הפקודה עשוי לכלול שם תיקיה, שם קובץ ספציפי או את שני הדברים גם יחד. לדוגמה, הביטוי C:*.BAT הוא נתיב הגישה אל כל הקבצים המאוחסנים בספריית השורש של כונן C ששמם כולל את הסיומת BAT.
תחביר הפונקציה $Dir הוא:

```
stringvar = Dir(path[, attributes])
```

איתור קבצים

אחד השימושים של הפונקציה $Dir הוא לשם בדיקה האם קובץ מסוים קיים. אם תנסה לפתוח קובץ שאינו קיים או לבצע גישה אל קובץ שאינו קיים, תתרחש שגיאה. תוכל למנוע שגיאות כאלו על ידי כך שתשתמש במשפט $Dir כדי לוודא שהקובץ קיים, לפני שתנסה לפתוח אותו, כמו בדוגמה הבאה:

```
If Dir$("C:\MYFILE.TXT") = " " Then
MsgBox "The file was not found. Please try again!"
End If
```

אם הקובץ המבוקש קיים, הפונקציה $Dir מחזירה את שם הקובץ ללא הנתיב המלא אליו ואם לא מאותר קובץ התואם לביטוי הכלול בקריאה לפונקציה מוחזרת מחרוזת ריקה. אם הקובץ אכן קיים, תוחזר המחרוזת myfile.txt. תוכל גם לפשט את הפעולה על ידי כתיבת פונקציה גנרית (כללית) שתחזיר ערך True אם הקובץ אכן קיים:

```
Public Function bFileExists(sFile As String) As Boolean
	If Dir$(sFile) <> " " Then bFileExists = True Else bFileExists = False
End Function
```

הפונקציה תוכל לשמש לבדיקת קיום קובץ ששמו יועבר אליה, כמו בדוגמה הבאה:

```
Dim sUserFile As String
sUserFile = InputBox$("Enter the file name: ")
If Not bFileExists(sUserFile) Then
	MsgBox "The file does not exist. Please try again."
	End
End If
```

קטע הקוד שהוצג בדוגמה יסיים את פעולת התוכנית אם הקובץ הרצוי לא יאותר, כדי למנוע שגיאות שעלולות להתעורר בשלבים מאוחרים יותר. דרך נוספת למנוע את הבעיה היא לבקש מהמשתמש להזין שם קובץ, עד שהוא יזין שם קובץ חוקי.

הצגת רשימות קבצים ותיקיות

הפונקציה $Dir משמשת גם להחזרה של רשימת הקבצים הכלולים בנתיב נתון. אם תשתמש במשפט Dir, כל שמות הקבצים הרלוונטיים יוצגו על המסך, אך עקב העובדה שהפונקציה $Dir מיועדת להחזיר רק מחרוזת אחת, תידרש להשתמש בלולאה ולאחזר שם קובץ יחיד בכל מחזור לולאה (תוכל גם להציג את רשימת הקבצים על ידי שימוש בפקד תיבת רשימה List Box - שהוא אחד מפקדי ברירת המחדל של Visual Basic, שנסקרו בפירוט בפרק 4).

נניח שבתיקיה C:\DATA מאוחסנים מספר סוגי קבצים עם סיומת BMP (סיומת המציינת קבצי מפת סיביות). הנתיב בו תשתמש כדי לאחזר את שמות הקבצים האלה על ידי שימוש בפונקציה $Dir יהיה C:\DATA*.BMP. תוכל להשתמש בקטע הקוד הבא כדי לאחזר את שמות הקבצים ולהציג אותם במסגרת תיבת רשימה.

```
Dim sNextFile As String
sNextFile = Dir$("C:\Data\*.BMP")
Do While sNextFile <> " "
	lstPictureList.AddItem sNextFile
	sNextFile = Dir$
Loop
```

בדוגמת הקוד האחרונה ראית שהנתיב לקובץ נכלל רק בקריאה הראשונה לפונקציה Dir$, הקריאות הבאות לפונקציה התבצעו ללא שימוש בארגומנטים, כדי לציין שיש להשתמש בנתיב שפורט בקריאה הקודמת. כאשר לא מאותרים יותר קבצים התואמים לתנאי החיפוש, Dir$ מחזירה מחרוזת ריקה ולולאת While מסתיימת.

אזהרה:

בעת שימוש בפונקציה Dir$ במסגרת לולאה, צא מהלולאה מייד לאחר שמוחזרת מחרוזת ריקה. אם תנסה לבצע קריאה נוספת, יתרחש מצב שגיאה

הפרמטר האופציונלי השני של הפונקציה Dir$ משמש להעברת תנאים נוספים, שתוכל להתבסס עליהם בעת בחירת קבצים. לדוגמה, השימוש בקבוע vbDirectory יגרום לכך שברשימה תוחזרנה רק תיקיות משנה הכלולות בנתיב הנתון. השימוש בקבוע vbVolume יגרום לכך שתוחזר **תווית אמצעי אחסון** (Volume Label) של הדיסק הנתון. הקבועים שבהם ניתן להשתמש במסגרת הקריאה לפונקציה מפורטים בטבלה 21.1 (תוכל גם להציג רשימת תיקיות על ידי שימוש בפפקד Directory List Box).

טבלה 21.1: קבועים ששולטים על אופן הפעולה של הפונקציה Dir$

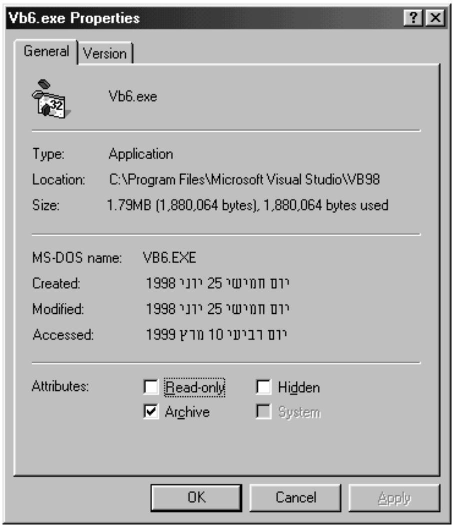
קבוע	ערך	פעולה
vbNormal	0	ערך ברירת מחדל
vbHidden	2	הצג קבצים נסתרים ברשימת הקבצים
vbSystem	4	הצג קבצי מערכת (System Files) ברשימת הקבצים
vbVolume	8	הצג את תווית אמצעי האחסון
vbDirectory	16	הצג תיקיות משנה ברשימת הקבצים
vbReadOnly	1	הצג קבצים לקריאה בלבד ברשימת הקבצים

הערה:

תוכל להשתמש בכמה קבועים במסגרת פנייה לפונקציה Dir$. לדוגמה, שורת הקוד הבאה תאתר את הקובץ IO.SYS שהוא קובץ מערכת (System), נסתר (Hidden) לקריאה בלבד (Read-only), במחשב המריץ Windows 95:

```
Debug.Print Dir$("C:\IO.SYS", vbHidden+vbSystem+vbReadOnly)
```

זכור שהקבוע vbHidden מתייחס ל**תכונות** (Attributes) הקובץ ולא לאפשרות של סייר Windows המשמשת להצגת קבצים נסתרים. את תכונות הקובץ תוכל להציג על ידי לחיצה ימנית על שם הקובץ ובחירה בפקודה Properties מתוך התפריט המקוצר שיוצג, כמתואר בתרשים 21.1, או על ידי שימוש בפקודה ATTRIB של DOS.



תרשים 21.1: תכונות קבצים (File Attributes) שמוצגות בסייר Windows יכולות לקבוע האם הפונקציה Dir$ תחזיר שם קובץ מסוים

פונקציות המבצעות פעולות על קבצים

בדומה לפונקציה Dir$, גם רוב הפונקציות האחרות של Visual Basic לביצוע פעולות על קבצים הן פשוטות לשימוש כמו הפקודות המקבילות ב-DOS, אך השימוש בהן כרוך במספר הגבלות. הפונקציות מפורטות בטבלה 21.2.

טבלה 21.2: תקציר הפונקציות של Visual Basic לביצוע פעולות על קבצים

פעולה	תחביר
העתקת קובץ	File Copy *source, dest*
מחיקת קובץ אחד או מספר קבצים	Kill *path*
שינוי שם קובץ	Name *oldname* As *newname*
יצירת תיקיה חדשה	MkDir *pathname*
מחיקת תיקיה ריקה	RmDir *pathname*
החלפת התיקיה הפעילה	ChDir *pathname*
החלפת הדיסק הפעיל	ChDrive *drive*

בסעיפים הבאים נפרט על מספר פונקציות שפורטו בטבלה.

העתקת קבצים

המשפט FileCopy מוגבל בכך שהוא אינו מאפשר להשתמש בתווי חיפוש כלליים לצורך הגדרת מספר קבצים בבת-אחת. הוא מאפשר להעתיק קבצים במחשב מקומי או ברשת, כמתואר בדוגמאות הבאות:

```
' The following line copies a file while changing it's name:
FileCopy "D:\My Documents\Hey Now.txt", "C:\DATA\TEST.TXT"

'The following lines use a network path for the source file:
Dim sDest As String
Dim sSource As String

sSource = "\\myserver\deptfiles\budget98.XLS"
sDest = "C:\DATA\BUDGET.XLS"
FileCopy sSource, sDest
```

משפט FileCopy גורם לדריסת קובץ היעד, אלא אם כן הוא מוגדר לקריאה בלבד או שהוא פתוח על ידי יישום אחר.

הערה:

בעת העתקת קבצים ברשת או באמצעות התקשרות בחיוג, כדאי להשתמש בפקודה On Error ל"לכידת" מצבי שגיאה. לדוגמה, מצבים שבהם הרשת אינה תקינה או שבהם משתמשים אחרים נעלו קבצים, עלולים לגרום לתוכנית שלך "לעוף" באופן בלתי צפוי. כדאי שתכלול בתוכנית שלך קטעי קוד ללכידת מצב השגיאה ולהצגת תיבת דו-שיח שתאפשר למשתמש לבחור האם לנסות לבצע שוב את הפעולה, כפי שתואר בפרק 10.

ראה, "שימוש בפקודה On Error", פרק 10.

טיפ:

בעת ביצוע פעולה הארוכה יותר משניות בודדות כדאי שתציג קובץ AVI, כפי שמערכת ההפעלה Windows נוהגת. יכולת זו תוארה בפרק 12.

מחיקת קבצים

Visual Basic גם מאפשרת לך למחוק קבצים על ידי שימוש במשפט Kill. משפט Kill מאפשר להשתמש בתווי חיפוש כלליים לצורך הגדרת פעולה על קבצים מרובים, כמו בדוגמה הבאה:

```
Kill "D:\NewDir\*.doc"
```

שינוי שמות קבצים

אופן פעולת המשפט Name של Visual Basic דומה לזה של הפקודה RENAME של DOS. אך פעולה זו פועלת רק על קובץ אחד ברגע נתון:

```
Name oldname As newname
```

במשפט Name ניתן להשתמש גם באופן הדומה לשימוש בפקודה MOVE של DOS, כדי לאחסן את הקובץ בנתיב שונה מזה של הקובץ המקורי:

```
' Move the file to a new directory
mkDir "D:\NewDir"
Name "C:\Windows\Desktop\TEST1.TXT" AS "D:\NewDir\TEST2.TXT"
```

בקטע הקוד האחרון הבחנת בוודאי במשפט mkDir, שמשמו משתמע שהוא משמש ליצירת תיקיה חדשה. המשפטים mkDir ו-rmDir משמשים לשם יצירת תיקיות חדשות וגריעת תיקיות קיימות, כפי שעושות פקודות DOS המקבילות להם, MD ו-RD.

הגדרת התיקיה הפעילה

בכל הדוגמאות שהובאו עד כה, הנתיב אל הקובץ כלל מציין כונן ותיקיה. אך אם התנסית בעבודה עם DOS סביר להניח שזכור לך שבכל רגע נתון אתה "ממוקם" בספריה מסוימת. לדוגמה, לאחר שתקליד CD \WINDOWS יתאפשר לך לבצע פעולות על קבצים המאוחסנים בתיקיה WINDOWS מבלי להידרש לציין את הביטוי C:\WINDOWS במסגרת הנתיב שבו תשתמש לצורך פנייה אל קבצים. המושג **ספריה פעילה** (Current Directory) תקף גם בעת עבודה ב- Visual Basic. השימוש במשפטים chDir ו-chDrive מאפשר לך לעבור בין תיקיות של דיסק מסוים ובין דיסקים, תוך הימנעות מהצורך לציין את הנתיב של הקבצים עבור כל פעולה בנפרד:

```
'Change to the desired directory and drive and rename a file
chDir "C:\Windows\Desktop"
chDrive "C:"
Name "TEST.TXT" As "TEST2.TXT"

'Delete a file in the current directory
chDrive "D:"
chDir "D:\DATA"
Kill "OLDDATA.DAT"
```

ביצוע פעולות מחיקת קבצים במצבים בהם לא ידוע לך מהי התיקיה הפעילה, כרוך בסיכון מסוים. למרבה המזל, Visual Basic מציעה לך פונקציה שתחזיר לך את שם התיקיה הפעילה, CurDir$. תחביר הפונקציה CurDir$ הוא:

```
stringvar = CurDir$( [Drive])
```

הערה:

אם תרצה לברר מי הדיסק הפעיל, תוכל להשתמש בפונקציה Left$ באופן הבא:

```
sDriveLetter = Left$(CurDir$( ), 1)
```

שימוש בפונקציה Shell להפעלת תוכניות נוספות

מדי פעם תידרש להפעיל תוכנית אחרת הפועלת בסביבת Windows על ידי שימוש בקוד Visual Basic. לדוגמה, תוכל להשתמש ב- Visual Basic לשם תזמון פעולות כגון העברת קבצים או מחיקת קבצים. תוכל לבצע פעולות אלו על ידי בדיקת השעה המוגדרת במערכת והפעלת תוכנית משנית מתאימה בשעה היעודה. את התוכניות האחרות תוכל להפעיל על ידי שימוש במשפט Shell. תחביר משפט Shell הוא:

```
DoubleVar = Shell(pathname[,windowstyle])
```

קטע הקוד הבא ישתמש במשפט Shell כדי להפעיל את היישום Notepad (פנקס הרשימות) שיציג קובץ בשם README.TXT:

```
Dim dTaskID As Double
dTaskID = Shell("Notepad D:\readme.txt", vbNormalFocus)
```

הפרמטר הראשון של הפונקציה Shell הוא הפקודה שאותה יש להפעיל והפרמטר השני (האופציונלי) הוא סגנון החלון שבמסגרתו יוצג היישום. הקבוע vbNormalFocus שבדוגמה מורה ל-Windows להפעיל את התוכנית בחלון רגיל שיקבל את מוקד הקלט. קבוע זה מגדיר שתי תכונות של החלון, סגנון (Style) וסוג מוקד (Focus). סגנון מגדיר את אופן הצגת החלון (Normal, Maximized, Minimized או Hidden). פרמטר המוקד מגדיר אם היישום יהפוך ליישום ב**קידמה** (Foreground - ברגע נתון רק יישום פעיל אחד יכול להיות במוקד הקלט). סגנון ברירת המחדל של החלון הוא חלון ממוזער שיוצג במוקד הקלט. השימוש בסגנון ברירת המחדל יגרום לכך שחלון היישום שיופעל על ידי משפט Shell לא יסתיר חלון של יישום אחר, אך הוא יופעל במוקד הקלט.

קל להפעיל יישומים נוספים על ידי שימוש במשפט Shell. אם יהיה עליך להפעיל תוכנית שתידרש להמתין לסיום פעולתה כדי שהקוד שלך יוכל להמשיך לפעול, תידרש לכתוב קוד שיהיה קצת יותר מורכב ויכלול מספר קריאות ל-Windows API. דוגמה של אופן פעולה כזה הובאה בפרק 20 **גישה ל-API של Windows**.

ראה, "המתנה לסיום פעולת תוכנית", פרק 20.

איתור קבצים תוך התייחסות ליישום שלך

בעבודה עם קבצים ב- Visual Basic לא מומלץ לקדד שמות קבצים ותיקיות בקידוד קשיח. יישומים מסחריים, כדוגמת Office, מאפשרים להחליף את תיקיית ההתקנה המהווה ברירת מחדל, לכל תיקיה הקיימת בדיסק שלך. גם אם תתקין את היישום שלך בתיקיה בעלת שם "קפריזי" כגון D:\JUNK123\, הוא יצליח לאתר את הקבצים שיידרשו לו לצורך פעולתו (כגון קבצי תבנית וקבצי תמונה שונים) ולפעול כרגיל.

היישום שלך גם יהיה מוצלח יותר ומקצועי יותר אם תפתח אותו באופן שיאפשר לו לפעול כשהוא מאוחסן בכל דיסק או בכל תיקיה שהמשתמש יבחר. אם ההתקנה שלך תכלול קבצים פרט לקובץ EXE (קובץ הביצוע) של היישום, תוכל לכלול ביישום מעט קוד שיאפשר לתוכנית לאתר את הקבצים באופן שקוף למשתמש.

בתרשים 21.2 מוצג מסך לבחירת תקליטורים. תוכנית זו מאפשרת למשתמש לבחור את התקליטור שיושמע, באמצעות לחיצה על תמונתו. התוכנית מיועדת לפעול מכונן תקליטורים, כך שנתיב הגישה לקובץ הביצוע עשוי להיות D:\TUNES\TUNES.EXE (בהנחה שהכונן שלך הוא כונן D:\). גם התמונות של עטיפות התקליטורים והנתונים הכלולים במסד הנתונים של התוכנית מאוחסנים בתיקיה TUNES\ של התקליטור.



תרשים 21.2: כאשר התוכנית שלך תפעל ממקום בלתי ידוע, כגון כונן רשת ממופה או כונן תקליטורים, תוכל להשתמש במאפיין App.Path כדי לאתר את קבצי העזר שיידרשו

אם תרצה שהתוכנית תוכל תמיד לאתר את התמונות הדרושות לה, ללא תלות באות המציינת את כונן התקליטורים המותקן במערכת, תוכל להיעזר במאפיין Path (נתיב) של אובייקט App (כלומר בביטוי App.Path), כפי שנעשה בדוגמה הבאה:

```
'Open database
Set db = OpenDatabase(App.Path & "\albums.mdb")
'Load image
Set Picture1 = LoadPicture(App.Path & "\Picture1.GIF")
```

מאפיין App.Path יחזיר את שם התיקיה שבה פועל היישום. בדוגמה הקודמת, ערך המאפיין היה D:\TUNES או כל שם תיקיה אחר שבה הותקן קובץ TUNES.EXE.

הערה:

אם אתה עובד בסביבת הפיתוח המשולבת של Visual Basic וכבר שמרת את הפרויקט, המאפיין App.Path יכיל את הנתיב לתיקיית הפרויקט שלך (התיקיה בה מאוחסן קובץ VBP של הפרויקט). אם טרם שמרת את הפרויקט, App.Path יכיל את התיקיה שבה מותקנת Visual Basic.

בוודאי שמת לב שבדוגמאות הקודמות הוספנו לוכסן הפוך (\) לפני שם הקובץ. המאפיין App.Path אינו מכיל את הלוכסן המסיים את שם הנתיב, פרט למקרים בהם הוא מחזיר את תיקיית השורש של דיסק. תוכל להסתייע בפונקציה Right$ כדי לוודא אם שם הנתיב שהוחזר מכיל לוכסן סיום או לא:

```
'Sets sAppPath (assumed to a public variable) for use later in the program
sAppPath = App.Path
If Right$(sAppPath, 1) <> "\" then sAppPath = sAppPath & "\"
 'Later in the program:
Dim sDBLocation As String
sDBLocation = sAppPath & "finance.mdb"
```

יש מפתחים שנוהגים להשתמש במשפט chDir כדי להפוך את התיקיה ששמה מאוחסן במאפיין App.Path לתיקיה הפעילה, בתחילת התוכנית. אני נמנע מלהשתמש במשפט זה מפני שהשימוש בו עלול להשפיע על תוכניות אחרות הפועלות בסביבת Windows.

טיפ:

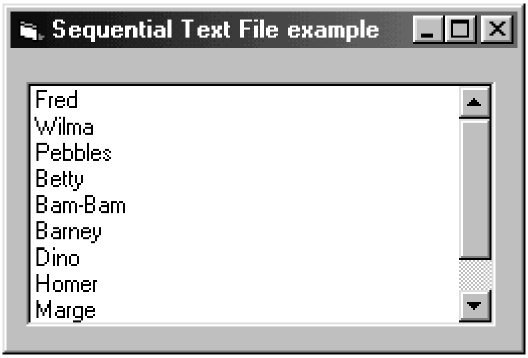
אם היישום שלך נעזר בקבצים רבים המאוחסנים בתיקיות שונות, כדאי שתצעד צעד נוסף קדימה ותשמור את מיקומי הקבצים בקובץ INI או במסד נתונים (עיין בסעיף "הבנת קבצי INI" בהמשך הפרק).

עבודה עם קבצי טקסט

במקרים מסוימים תידרש לאחסן נתונים ולאחזר נתונים, אך מבלי להידרש לעוצמה של מסד נתונים (וזאת מבלי להתייחס לקוד הנוסף, קבצי הגדרת התצורה וקבצי העזר שנדרשים לשם עבודה עם מסד נתונים). במקרים כאלה, **קבצי טקסט** (Text Files) יהוו את הפתרון הרצוי. בסעיף זה תלמד על הסוג הפשוט ביותר של קובץ טקסט **קובץ סדרתי בעל מבנה חופשי** (Free-form Sequential Text File). מבנה סדרתי הוא מבנה שבו הגישה אל בתים בקובץ נעשית על-פי סדר הופעתם בקובץ, מבלי להידרש לעבור אל מקום מסוים בקובץ. מבנה חופשי פירושו שמבנה הקובץ לא הוגדר מראש, והשליטה על מבנה הקובץ נתונה במלואה בידי המשתמש.

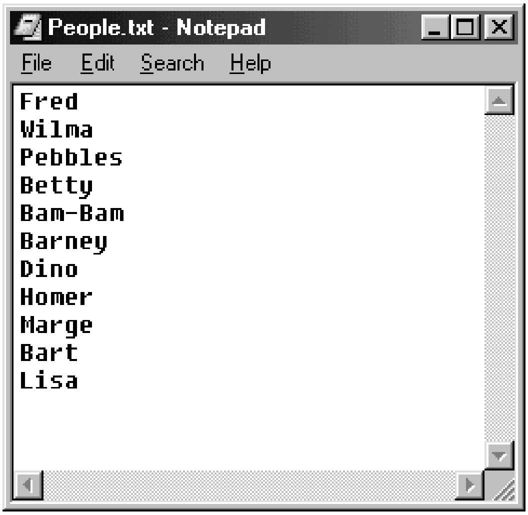
קבצי טקסט סדרתיים

נניח שאתה עובד עם טופס שמאפשר למשתמש לבחור שם של איש מכירות, כמו זה המוצג בתרשים 21.3. אתה תידרש לטעון את שמות אנשי המכירות אל תוך תיבת רשימה, כדי לאפשר למשתמשים לבחור מתוכה שם. את הפעולה של טעינת השמות לתיבת הרשימה תוכל לבצע על ידי שימוש בסדרת משפטי AddItem במסגרת שגרת Form_Load של הטופס. אך לא כדאי לך לנהוג כך מפני שנתונים אלה יקודדו בקידוד קשיח. בכל פעם שיחול שינוי ברשימת אנשי המכירות של החברה תידרש להדר את התוכנית מחדש. פתרון נוח יותר לבעיה הוא שימוש בקובץ טקסט שיכיל את שמות אנשי המכירות. התוכנית תקרא את תוכן קובץ הטקסט ותכניסו לתיבת הרשימה.



תרשים 21.3: תיבת הרשימה מאוכלסת בתוכנו של קובץ הטקסט

תוכל ליצור את הקובץ עצמו על ידי שימוש בעורך טקסט כדוגמת Notepad, תוך רישום שמו של כל איש מכירות בשורה נפרדת, כמתואר בתרשים 21.4.



תרשים 21.4: פשוט מאוד לערוך נתונים המאוחסנים בקבצי טקסט סדרתיים

התהליך המיושם לצורך כך הוא פשוט וכל אחד יכול לערוך את הקובץ, גם מי שאין בידו תוכנה לעבודה עם מסדי נתונים. לדוגמה, המזכירה תוכל לנהל את הקובץ שיישמר בשרת והיישום יוכל להעתיק את הגירסה העדכנית של הקובץ כאשר הוא יאותחל. אם אתה בלאו הכי משתמש במסד נתונים, תוכל לאחסן את השמות בטבלה, אך תוכל גם להשתמש בקובץ טקסט פשוט לשם אחסון הטבלה.

הערה:

המושג של קבצים משותפים ברשת מתייחס גם לקבצי מסדי נתונים, וכן לכל קובץ אחר העשוי להועיל למישהו מבין המשתמשים ברשת.

קריאה מקובץ טקסט סדרתי

כעת, כאשר ידוע לך עד כמה קל ליצור קבצי טקסט סדרתיים, תוכל לכתוב קוד שיקרא נתונים מקובץ כזה. אחת הדרכים הפשוטות לעיבוד קובץ טקסט סדרתי היא על ידי קריאת כל שורה בנפרד. הפעולות הדרושות לשם קריאה מהקובץ המכיל את שמות אנשי המכירות הן פשוטות מאוד:

1. פתח את הקובץ לקריאה.
2. קרא שורה מהקובץ ואחסן אותה במשתנה.
3. צרף את תוכן המשתנה לתיבת הרשימה.
4. בצע את צעדים 2 ו-3 עבור כל שורה בקובץ.
5. סגור את הקובץ.

הקוד שישמש למילוי תיבת הרשימה, שיידון בסעיפים הבאים, מוצג בתרשים 21.1.

תוכנית 21.1: LISTFILL.VBP - מילוי תיבת רשימה מקובץ טקסט.

```
Sub FillListBox()
        Dim sTemp As String
        lstPeople.Clear
        Open "C:\DATA\PEOPLE.TXT" For Input As #1
        Do While Not EOF(1)
                Line Input #1, sTemp
                lstPeople.AddItem sTemp
        Loop
        Close #1
End Sub
```

לפני שתוכל לכתוב או לקרוא נתונים מהקובץ, תידרש לפתוח את הקובץ על ידי משפט Open. משפט זה יקשר את השם הפיסי של הקובץ עם **מספר קובץ** (File Number). מספר הקובץ הוא ערך מספרי שלם המשמש לזיהוי הקובץ בקוד Visual Basic:

```
Open "C:\DATA\PEOPLE.TXT" For Input As #1
```

הערה:

בקטע הקוד שהוצג כאן, מספר הקובץ הוא 1. אך אם תפתח ותסגור קבצים רבים במהלך פעולת התוכנית, לא כדאי שתשתמש במספר מפורש. כדאי שתשתמש בפונקציה FreeFile שתחזיר את מספר הקובץ הפנוי הבא. כמו בדוגמה הבאה:

```
Dim nFile As Integer
nFile = FreeFile
Open "C:\MYFILE.TXT " for Input As #nFile
```

לאחר שתסיים להשתמש בקובץ, כדאי שתסגור אותו באמצעות משפט Close (ראה תוכנית 21.1) כדי לפנות את מספר הקובץ לשימוש על ידי קבצים אחרים.

בנוסף לכך שהיא מקשרת בין מספר הקובץ לשם הקובץ, משפט Open גם מורה ל- Visual Basic מהו האופן בו אתה מתכוון להשתמש בקובץ מסוים (משפט Open תומך באפשרויות רבות, שמידע אודותיהן תוכל למצוא בקובץ העזרה).

טיפ:

לפני שתפתח קובץ לקריאה, כדאי שתיעזר בפונקציה Dir$ כדי לוודא שהוא קיים.

מילת המפתח Input מציינת שהקובץ נפתח לקריאה סדרתית, אופן פתיחה אשר מאפשר לעבור על-פני הרשומות, על-פי סדר הופעתן בקובץ, תוך תנועה קדימה. פעולת הקריאה מעבירה את מצביע הקובץ אל הרשומה הבאה, באופן אוטומטי. קטע הקוד שהוצג בתוכנית 21.1 משתמש במשפט Line Input הכלול בלולאת Do While לצורך קריאת הנתונים. משפט Line Input הראשון יקרא את השורה הראשונה בקובץ, משפט Line Input השני יקרא את השורה השנייה של הקובץ וכך הלאה. השורות בקובץ מופרדות על ידי סימני End of Line (סוף שורה). סימן End Of Line בסביבת Windows מורכב מתו החזרת גררה (Carriage Return) משולב עם תו הזנת שורה (Line Feed).

תחביר משפט Line Input הוא:

```
Line Input #filenumber, variablename
```

כאשר *Filenumber* מכיל את מספר הקובץ הפתוח ו-*Variablename* הוא משתנה מחרוזת או משתנה מסוג Variant. אם תנסה לקרוא יותר שורות מכפי שקיימות בקובץ, תתרחש שגיאה, על-כן כדאי שתיעזר בפונקציה EOF (End Of File - סוף קובץ) כדי לבדוק אם הגעת לסוף הקובץ, בטרם תבצע ניסיון נוסף לקריאה מהקובץ.

לאחר שתפתח את הקובץ יתאפשר לך לבחור מבין כמה שיטות לקריאה לצורך קריאה מתוכו. בדוגמה שלנו שמו של איש המכירות הוא הנתון היחיד הכלול בכל שורה, כך שלא תידרש לבצע פעולות נוספות לצורך עיבוד משתנה המחרוזת. אך במקרים מסוימים תרצה שכל פעולת קריאה לא תקרא שורה שלמה, או יותר מנתון אחד מכל שורה. במקרים כאלה יהיה עליך להשתמש במשפט Input# או בפונקציה Input.

משפט #Input מיועד לקריאת נתונים המופרדים בתווים מסוימים. לדוגמה, השורה הבאה הלקוחה מתוך קובץ טקסט מכילה שלושה פריטי מידע נפרדים: מחרוזת, מספר ותאריך. הנתונים מופרדים בפסיקים, בגרשיים ובתווי #.

```
"Test", 100, #1998-01-01#
```

שורת הקוד הבאה תקרא כל אחד מהנתונים שהוצגו בשורה הקודמת אל תוך משתנה מהסוג המתאים:

```
Input # 1, stringvar, intvar, datevar
```

זכור שהפונקציה # Input מחפשת את תווי ההפרדה האלה, כך שיהיה עליך להקפיד שמבנה משפט # Input יתאים למבנה הקובץ.

הפונקציה Input היא אמצעי נוסף לקריאת נתונים. הפונקציה Input מאפשרת להגדיר את מספר התווים שייקראו מהקובץ, כפי שנעשה בדוגמה הבאה:

```
'Reads five Characters from file number 1
s = Input(5,#1)
```

כעת נשווה את האופנים בהם ניתן להשתמש בשיטות השונות שהוצגו כדי לעבד שורה מסוימת בקובץ:

```
'Assume our file has the following line repeated in it: "This is a test string."
Dim s As String
Line Input #1, s
' s contains the entire string, including quotes
Input #1, s
's contains the string without quotes
s = Input(5,#1)
' s Contains the first 5 characters ("This)
```

כתיבה בקובץ טקסט סדרתי

קובץ יומן (Log File) הוא אחד מאופני השימוש הנפוצים בקבצי טקסט סדרתיים. אני משתמש ב- Scheduler Application שמפעיל תוכניות ומבצע עדכונים במסד נתונים ואני ממעט להשתמש במחשב שעליו פועל Scheduler Application, אך אני נוהג להתחבר אל המחשב הזה דרך הרשת ולעיין בקובץ היומן כדי לוודא אילו עדכונים הושלמו עד כה.



טיפ:

קבצי טקסט סדרתיים מאפשרים ליצור קבצי אצווה (Batch Files), תסריטי FTP (FTP Scripts) וכן סוגים נוספים של קבצי טקסט בעלי מבנה פשוט, באופן ממוכן.

תוכנית 21.2 מציגה תת-שיגרה בשם LogPrint שאותה תוכל לצרף לתוכנית שלך לצורך ניהול קובץ יומן של הודעות שגיאה. תת-השיגרה תרשום את הודעות השגיאה בקובץ היומן, כאשר כל הודעה מלווה בתאריך.

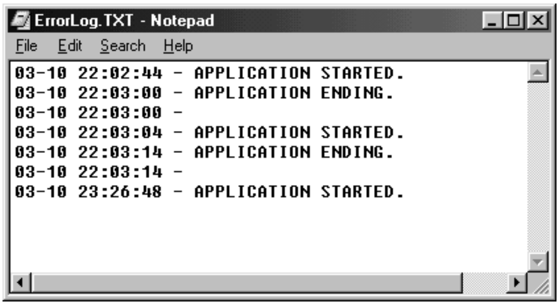
תוכנית 21.2: LOGPRINT.VBP - שימוש בקובץ פלט סדרתי לבניית **יומן יישום**.

```
Sub LogPrint(sMessage As String)
	Dim nFile As Integer
	nFile = FreeFile
	Open App.Path & "\ErrorLog.TXT" For Append Shared As #nFile
	Print #nFile, Format$(Now, "mm-dd hh:mm:ss") & " - " & sMessage
	Close #nFile
End Sub
```

לתת-שיגרה זו תוכל לקרוא מתוך שיגרה לטיפול בשגיאות, או כדי להודיע על התרחשות אירוע מסוים במסגרת התוכנית:

```
LogPrint "The Database was successfully opened."
```

תוכל לעיין בקובץ היומן בעזרת עורך טקסט, כמתואר בתרשים 21.5.



תרשים 21.5: תוכל לאפשר ליישום לנהל קובץ יומן על ידי הוספת שורות קוד מעטות

הסעיף הקודם השתמש במילת המפתח Input כדי לפתוח את הקובץ לקריאה (ראה תוכנית 21.1). כתיבת נתונים מתבצעת על ידי פתיחת הקובץ לכתיבה סדרתית, אך במקום להשתמש במילת המפתח Output יהיה עליך להשתמש במילת המפתח Append. מצא את ההבדלים בין שתי שורות הקוד הבאות:

```
'Append mode - adds to an existing file or creates a new one
Open "ErrorLog.TXT" for Append as #1

'Output Mode - always creates a new file, erases any existing information
Open "ErrorLog.TXT" for Output as #1
```

אופן העבודה Append גורם לכך שהנתונים שמוספים לקובץ מוספים לאחר הנתונים הקיימים בו. זהו אופן הפעולה המתאים ביותר לניהול קבצי יומן, מפני שבעת ניהול קבצים כאלה תרצה שנתונים חדשים יתווספו לקובץ הקיים. פתיחת קובץ תוך שימוש

באופן העבודה Output יגרום לכך שנתונים הקיימים בקובץ יימחקו. שני אופני הפעולה האלה גורמים לכך שמשפט Open ייצור קובץ חדש, אם לא קיים קובץ עם השם הכלול בו. לאחר שתפתח קובץ לכתיבה, תוכל להשתמש בכמה משפטים שונים לצורך כתיבת הנתונים בו. המשפטים #Print ו-#Write, שיתוארו בסעיף הבא, יאפשרו לך לעצב את הנתונים שתכתוב בקובץ באופנים שונים.

שימוש במשפטים Print ו-Write

אופן פעולת משפט #Print דומה מאוד לזה של השיטה Print שתוארה בפרק 20, **גישה ל-API של Windows**, פרט לכך שבמקום שפלט משפט יופנה אל אובייקט המוצג על גבי המסך, הוא מופנה אל קובץ פתוח. למעט במקרים שבהם נעשה שימוש בנקודה-פסיק או בתו הפרדה אחר בסוף השורה, מוסף תו מעבר שורה לאחר כל פריט שמודפס. תחביר משפט #Print הוא:

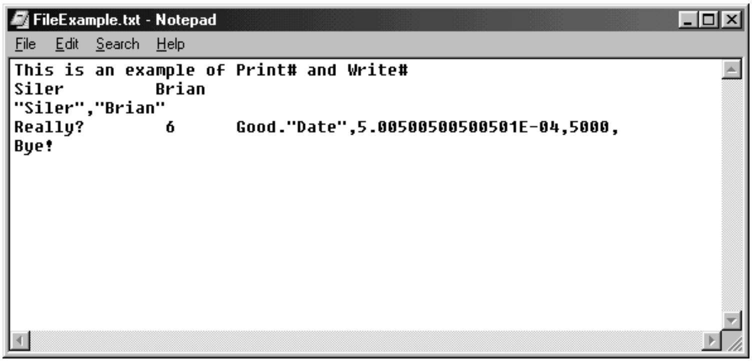
Print *#filenum, expressions*

משפט אחר, משפט #Write עובד כמו משפט #Print, אך מוסיף מפרידים ותווי הפרדה באופן אוטומטי. תחביר משפט #Write הוא:

Write *#filenum, expressions*

#Write הוא משפט המיועד לפעול בשילוב עם המשפט #Input שתואר בסעיף "קריאה מקבצי טקסט סדרתיים". להלן מובאות מספר דוגמאות של שימוש בשני המשפטים, שהקובץ שנוצר על ידי השימוש בהם מוצג בתרשים 21.6.

```
Print #1, "This is an example of Print# and Write#"
Print #1, "Siler", "Brian"
Write #1, "Siler", "Brian"
Print #1, "Really?", 2*3;Spc(5);"Good.";
Write #1, "Date", 1/1/1998,1000*5;
Print #1 vbCrLf & "Bye!"
```



תרשים 21.6: משפט #Print מאפשר שליטה טובה יותר על מבנה הקלט שיוצג ואילו משפט #Write מקל על אחזור הנתונים על ידי שימוש בתווים מפרידים

קבצים לגישה אקראית - הגדרת מבני קבצים

לקבצים סדרתיים אין מבנה מסודר כלשהו. למעשה, ניתן לומר שמבנה הקבצים הסדרתיים מוגדר על ידי הקוד שקורא את הקובץ ולא על ידי הקובץ עצמו. בסעיף הקודם למדת שעובדה זו מהווה גורם מועיל מבחינת קריאות הקבצים, אך היא מגבילה את מבחר הפונקציות שבהן יכולים מפתחים להשתמש לצורך קריאה מהקובץ. לדוגמה, Visual Basic אינה כוללת פונקציה שמאפשרת לדלג אל רשומה מסוימת בקובץ סדרתי, מפני שהיא אינה מכירה את מבנה הקובץ. אחת הדרכים שבהן ניתן להגדיר מבנה מסוים עבור קובץ היא על ידי אחסון מבני רשומות (Record Structures) שהוגדרו על ידי המפתח, במקום אחסון של מחרוזות פשוטות. השימוש במבנים כאלה מאפשר לפתוח את הקובץ תוך שימוש ב**מצב אקראי** (Random Mode), שאינו מגביל את המשתמש לגישה סדרתית.

יצירת סוג רשומה

סוגי רשומות מותאמים אישית (Custom Record Types) הם למעשה סוגי נתונים המוגדרים על ידי המשתמש (User-Defined Data Types). הגדרת סוגי נתונים המוגדרים על ידי המשתמש מתבצעת באמצעות משפט Type. הצהרות על סוגי נתונים כלולות בקטע General Declarations (ההצהרות הכלליות) של חלון הקוד. קטע הקוד הבא ישתמש במשפט Type לצורך הצהרה על סוג הנתונים Employee:

```
Private Type Employee
        EmpID As Integer
        LName As String * 30
        FName As String * 20
        Title  As String * 20
End Type
```

ניתן לבצע גישה אל נתונים מהסוג המותאם אישית על ידי שימוש בתו נקודה (Dot Notation) כמו בדוגמה הבאה:

```
Dim Emp1 As Employee
Emp1.Fname = "Joe"
Emp1.Lname = "Smith"
Emp1.Title = "Chicken Plucker"
Emp1.EmpID = "12345"
```

הגישה לשדות סוג נתונים מותאם אישית דומה לגישה למאפייני אובייקט, שתוארה בפרק 4, **שימוש בפקדי ברירת המחדל של Visual Basic**.

פתיחת קובץ לגישה אקראית

ההבדל העיקרי בין פתיחת קובץ סדרתי לבין פתיחת קובץ לגישה אקראית, הוא שבעת פתיחת קובץ לגישה אקראית תידרש לכלול במשפט Open הגדרה של אורך הרשומה. את אורך הרשומה של סוג שהוגדר על ידי המשתמש ניתן לברר על ידי שימוש בפונקציה Len על נתון מהסוג הרצוי, כמו בדוגמה הבאה:

```
Dim emp1 As Employee
Open "D:\EMPINFO.DAT" for Random As #1 Len = Len(emp1)
```

שורת קוד זו פותחת את הקובץ EMPINFO.DAT לגישה אקראית. קטע Len= של משפט Open מורה ל- Visual Basic להניח שפעולות הקריאה והכתיבה הבאות שתתבצענה על הקובץ הזה, תתייחסנה לרשומות הזהות באורכן למבנה emp1.

הוספת רשומות בעזרת משפט Put

לאחר שתפתח קובץ לגישה אקראית, תוכל להשתמש במשפט Put לשם אחסון רשומות בקבצים. תחביר משפט Put הוא כדלקמן:

```
Put filenumber,[recnumber], variablename
```

קטע הקוד הבא ישתמש במשפט Put במסגרת לולאת For לצורך כתיבת חמש רשומות מסוג Employee בקובץ #1:

```
For i = 1 To 5
	emp1.LName = InputBox$("Enter Last Name")
	emp1.Fname = InputBox$("Enter First Name")
	emp1.Title = InputBox$("Title")
	emp1.EmpID = i
	Put #1, emp1
Next i
```

ממשפט Put הושמט הפרמטר האופציונלי recnumber שמגדיר את המקום בקובץ שבו תיכתב הרשומה החדשה. במקרים בהם לא נתונה הגדרה עבור פרמטר זה, הרשומה החדשה נכתבת במקום שאליו מכוון מצביע הקובץ.

אחזור רשומות בעזרת משפט Get

משפט Get ישמש אותך לאחזור רשומות מקובץ אקראי. משפט Get יכול לשמש לקריאת נתונים בחזרה לתוך רשומה מהסוג שהגדרת, כפי שניתן לראות בדוגמה הבאה שתקרא את רשומה מספר 4 לתוך המשתנה emp1 ותציג את הערך המאוחסן בשדה Title של הרשומה:

```
Get #1, 4, emp1
MsgBox "Employee title is " & emp1.Title
```

תחביר משפט Get דומה לזה של משפט Put :

```
Get filenumber, [recnumber], variablename
```

אם תשמיט את הפרמטר recnumber מהקריאה למשפט Get, משפט Get יפעל כמו משפט Line Input, כלומר הוא יקרא את שורות הקלט בהתאם לסדר הופעתן בקובץ.

שימוש במשפט Seek לגישה אקראית

תוכל להשתמש במשפט Seek כדי לעבור בין רשומות בקובץ. משפט Seek מקבל שני פרמטרים, מספר הקובץ ומספר הרשומה, כמו בדוגמה הבאה :

```
Seek #1, 3
```

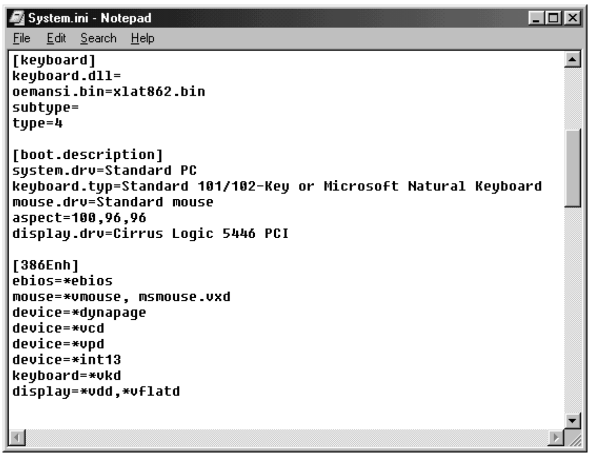
שורת קוד זו תגרום לכך שמשפט Put או Get הבא שיבוצע יתייחס לרשומה מספר 3.

קבצי INI

קבצי INI משמשים לאחסון נתוני תוכניות והגדרות משתמשים. כעיקרון, קובץ INI הוא קובץ טקסט בעל מבנה פשוט, אשר מאפשר לך לשמור נתונים מסוימים ולאחזר אותם. הסיומת INI היא קיצור של **אתחול** (Initialization). אחסון נתונים בקבצי INI מונע את הצורך לקדד אותם בקידוד קשיח במסגרת התוכנית שלך. כך מתאפשר לך לשנות ערכי פרמטרים מבלי להידרש להדר שוב את התוכנית. בסעיפים הבאים נסקור כמה שימושים אפשריים של קבצי INI ונתאר את הפעולות הדרושות לשם שימוש בהם במסגרת תוכנית.

הבנת קבצי INI

מבנה קבצי INI הוא פשוט. ניתן לעיין בהם ולבצע שינויים על ידי שימוש בעורך טקסט כלשהו, כגון Notepad. בתרשים 21.7 מוצג קובץ INI לדוגמה.



תרשים 21.7: קבצי INI יכולים לשמש לאחסון הגדרות ונתונים אחרים באופן מסודר

שלושת המרכיבים של קבצי INI נקראים **מקטעים** (Sections), **מפתחות** (Keys) ו**ערכים** (Values) (בטרמינולוגיה של Microsoft כל מקטע נקרא גם Application - יישום, זכר לימים שבהם כל יישום אחסן את ההגדרות שלו בקובץ WIN.INI). מרכיבי קובץ INI יפורטו בטבלה 21.3.

טבלה 21.3: מרכיבי קובץ INI

מרכיב	תיאור
Section	שם התחום בסוגריים מרובעים ([]) תחתיו מקובצת סדרת מפתחות וערכים
Key	מחרוזת ייחודית. לאחר המפתח יוצבו סימן שווה (=) וערך. מפתח נדרש להיות ייחודי במקטע בו הוא כלול
Value	הנתונים הממשיים שקשורים למפתח מסוים בקובץ INI. מקטע ומפתח משמשים יחד לשם קריאה או כתיבת ערך

סדר הופעת המקטעים בקובץ אינו חשוב, משום ששילוב שם מקטע ושם מפתח מצביע (או לפחות אמור להצביע) על ערך יחיד. בקבצי INI נקודה פסיק (;) מסמנת הערה, מערכת ההפעלה מתעלמת ממפתחות ומערכים שמופיעים לאחר נקודה פסיק. לדוגמה:

```
[Settings]
DBLocation=P:\USERINFO.MDB
;DBLocation=D:\CODE\TEST.MDB
```

בדוגמה זו קל מאוד לעבור משימוש במסד נתונים מקומי המשמש לצורך תהליך הפיתוח, לשימוש במסד הנתונים האמיתי של היישום. עליך רק להפוך את השורה שמצביעה על מסד הנתונים שאינך מעוניין להשתמש בו להערה.

שימוש בקבצי INI ב- Visual Basic

אחת הסיבות שבעטיין קבצי INI נחשבים לפשוטים לשימוש היא שהם חוסכים ממך את הצורך להתייחס לפעולות הבאות: יצירת קובץ, פתיחת קובץ ואיתור השורה הרצויה בקובץ. לפני שתוכל להשתמש בקובץ INI תידרש להצהיר על שתי פונקציות של Windows API ולכתוב פונקציות מעטפת (Wrapper Function) עבורן (מידע נוסף אודות פונקציות הכלולות ב- Windows API תוכל למצוא בפרק 20 **גישה ל-API של Windows**. הוסף לתוכנית שלך מודול חדש והוסף את הקוד הכלול בתוכנית 21.3 לקטע General Declarations שלו.

טיפ:

כדאי לך לבנות ספריית פונקציות מועילות כגון אלו במודול נפרד (שתוכל לקרוא לו UTILITY.BAS, או שם דומה), שבו תוכל להשתמש בקלות כחלק מכמה פרויקטים.

תוכנית 21.3: INIFUNCS.VBP - שימוש בקבצי INI במסגרת תוכנית.

```
'API DECLARATIONS
Declare Function GetPrivateProfileString Lib "kernel32" Alias "GetPrivateProfileStringA" _
        (ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal _
        lpDefault  As String, ByVal lpReturnedString As String, ByVal nSize As _
        Long, ByVal lpFileName As String) As Long
Declare Function WritePrivateProfileString Lib "kernel32" Alias _
        "WritePrivateProfileStringA" (ByVal lpApplicationName As String, ByVal _
        lpKeyName As Any, ByVal lpString As Any, ByVal lpFileName As String) _
        As Long

Public Function sGetINI(sINIFile As String, sSection As String, sKey As String, sDefault _
        As String) As String

    Dim sTemp As String * 256
    Dim nLength As Integer

    sTemp = Space$(256)
    nLength = GetPrivateProfileString(sSection, sKey, sDefault, sTemp, 255, sINIFile)
    sGetINI = Left$(sTemp, nLength)

End Function

Public Sub writeINI(sINIFile As String, sSection As String, sKey As String, sValue As _
        String)

    Dim n As Integer
    Dim sTemp As String

    sTemp = sValue

    'Replace any CR/LF characters with spaces
    For n = 1 To Len(sValue)
        If Mid$(sValue, n, 1) = vbCr Or Mid$(sValue, n, 1) = vbLf  Then _
            Mid$(sValue, n) = " "
    Next n

    n = WritePrivateProfileString(sSection, sKey, sTemp, sINIFile)

End Sub
```

לאחר שתוסיף ליישום את הקוד המובא בתוכנית 21.3, תוכל להשתמש בפונקציה sGetINI ובתת-שיגרה writeINI לשם כתיבה וקריאה נוחה של קבצי INI מהם ואליהם. קטע הקוד הבא ידגים לך כיצד לאחזר הגדרות מקובץ INI לצורך שימוש בעת אתחול תוכנית.

תוכנית 21.4: INIFUNCS.VBP - שימוש בקבצי INI לאחסון הגדרות של תוכניות.

```
Sub InitProgram()

    Dim sINIFile As String
    Dim sUserName As String
    Dim nCount As Integer
    Dim i As Integer

    'Store the location of the INI file
    sINIFile = App.Path & "\MYAPP.INI"

    'Read the user name from the INI file
    sUserName = sGetINI(sINIFile, "Settings", "UserName", "?")

    If sUserName = "?" Then
        'No user name was present - ask for it and save for next time
        sUserName = InputBox$("Enter your name please:")
        WriteINI sINIFile, "Settings", "UserName", sUserName
    End If

    'Fill up combo box list from INI file and select the user's last chosen item
    nCount = CInt(sGetINI(sINIFile, "Regions", "Count", 0))
    For i = 1 To nCount
        cmbRegn.AddItem sGetINI(sINIFile, "Regions", "Region" & i, "?")
    Next i

    cmbRegn.Text = sGetINI(sINIFile, "Regions", "LastRegion", cmbRegn.List(0))

End Sub
```

קטע הקוד המובא בתוכנית 21.4 בודק תחילה בקובץ INI לצורך איתור שם משתמש. על ידי הגדרת התו '?' כערך ברירת מחדל עבור שם המשתמש, תוכל לבדוק האם בקובץ כלול כבר שם משתמש ואם לא, תוכל להציב בקשה מהמשתמש להגדיר אותו. היכולת להשתמש בערך ברירת מחדל עבור הפרמטר היא חשובה מאוד, מפני שניתן להשתמש בה גם בעת התייחסות אל קובץ שאינו קיים.

הקטע הבא של דוגמת הקוד משמש לקריאת מספר האזורים מקובץ INI וייעזר בערך שייקרא כדי להציב כל אזור בתיבה משולבת. השורה האחרונה של הקטע מציבה במאפיין הטקסט של התיבה המשולבת ערך שנקרא מקובץ INI תוך שימוש בפריט הראשון של הרשימה כערך ברירת מחדל.

הערה:

זכור כי writeINI היא תת שיגרה, כך שאין להשתמש בסוגריים במסגרת הקריאה אליה.

הפעלת התוכנית שלך תוך שימוש בקבצי INI, באופן שהודגם בתוכנית 21.4, יאפשר לך לבצע שינויים קלים בתוכנית, במהירות וביעילות מבלי שתידרש להדר אותה שוב.

מכאן...

בפרק זה למדת על כמה מבין הפונקציות לטיפול בקבצים המובנות ב- Visual Basic וכן למדת כיצד להשתמש בקבצי INI במסגרת יישומים. בעת עבודה ב- Visual Basic תוכל להעתיק, למחוק, לשנות שמות קבצים ואף לאחסן בהם נתונים תוך שימוש במיגוון מבני אחסון. לשיטות שתוארו בפרק זה אין אומנם עוצמה רבה כפי שיש למנגנוני ניהול מסדי נתונים, אך הן מותאמות היטב לביצוע מטלות פשוטות שאינן כרוכות בתקורה רבה.

מידע נוסף אודות נושאים שנדונו בפרק זה, תוכל למצוא גם בפרקים הבאים:

- אם תרצה ללמוד עוד אודות שיטות מתקדמות לאחסון נתונים תוכל לעיין בפרק 24 **יסודות מסד הנתונים**.
- מידע נוסף אודות פונקציות Windows API תוכל למצוא בפרק 20 **גישה ל-API של Windows**.
- מידע נוסף אודות גישה לקבצים מתוך דפדפן Web תוכל למצוא בפרק 30 **שימוש ב-VBScript**.

22

שימוש ב-OLE לשליטה על יישומים אחרים

מה בפרק זה?

- עבודה עם אובייקטים של Word
- עבודה עם Excel
- שימוש בפקד המכולה OLE

OLE, שפירושו קישור והטבעת אובייקטים, הוכנס כאמצעי לשילוב מוצרים ממשפחת Microsoft Office. יישומים תומכי OLE מאפשרים למשתמש להפעיל יישום אחד מתוך אחר ללא צורך לצאת מהיישום המקורי ולעזוב את הממשק שלו. Microsoft Office היא דוגמה מצוינת של קבוצת תוכניות עצמאיות המשתמשות בטכנולוגיית OLE לעבוד יחד. מאפיין זה הופך קבוצת יישומים ליותר מאשר חבילת תוכנות ארוזות יחדיו.

לדוגמה, על ידי שימוש בטכנולוגיית OLE ניתן לשבץ גיליון Excel במסמך Word. משתמש העובד עם מסמך Word יכול לערוך גיליון בפשטות על ידי לחיצה כפולה עליו. פעולה זו תפתח את תכונות העריכה של Excel בזמן העבודה בתוכנית Word.

היתרונות למשתמש הם בכך שניתן לערוך מסמכים שונים מתוך ממשק אחיד וקבוע. היתרון למפתחים הוא שיישומי Office מכילים קבוצה גדולה של אובייקטי OLE (או ActiveX) הנגישים למשתמשים מתוך תוכניות Visual Basic.

כאשר משתמשים באובייקטי OLE של Office מתוך Visual Basic, למעשה "שולטים מרחוק" על Word או Excel. פירוש הדבר הוא שהתוכנה חייבת להיות מותקנת במחשב המשתמש והיישום מתנהג השינויים נעשים מתוכו. היישום עולה ובהתאם לצורת כתיבת קוד Visual Basic, המשתמש יכול לצפות או לתקשר עם היישום. מסיבות אלו אוטומציית OLE יכולה להיות איטית אך שימושית בנסיבות מסוימות.

להלן מספר יתרונות לשימוש באוטומציית OLE:

- **תצוגת דוחות מקצועיים** באמצעות Word ו-Excel ניתן להדפיס דוחות מקצועיים ומרשימים. על ידי שליטה מרחוק על Word ו-Excel מ- Visual Basic, ניתן בקלות להדפיס דוחות תוך ניצול תכונות העריכה המתקדמות של תוכנות אלו.
- **יצירה אוטומטית של מסמכים** ניתן ליצור גליונות עבודה ומסמכים על ידי שימוש בקוד Visual Basic. לדוגמה, ניתן לרכז כל שבוע הצעות עבודה מתוך מאגר נתונים, ולשלוח אותם למועמדים המתאימים באמצעות e-mail.
- **שליפת מידע** ניתן לשלוף מידע מתוך גיליון אלקטרוני באמצעות אובייקטי OLE ולאחסנו במאגר נתונים.

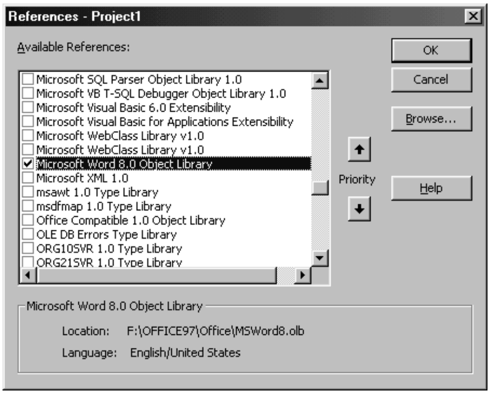
בפרק זה נבחן מספר דרכים להשתמש בתוכנות Word ו-Excel מתוך Visual Basic. למרות שהדוגמאות מכילות קוד Visual Basic סטנדרטי, קח לתשומת לבך כי יישומי Office מכילים למעשה קוד Visual Basic בתוכם ומרבית הקודים שמתועדים כאן ניתן לשלב כקוד VBA בתוך מסמך או גיליון.

עבודה עם אובייקטי Word

היכולת לשלוט במרכיב Microsoft Word מתוכנית Visual Basic יכולה להוות ערך מוסף משמעותי ליישום. המשתמשים יעריכו את היכולת לערוך מסמך Word ללא צורך לצאת מתוכנה מסוימת. בפרק זה תלמד כיצד להוסיף יכולות Word לתוכנית.

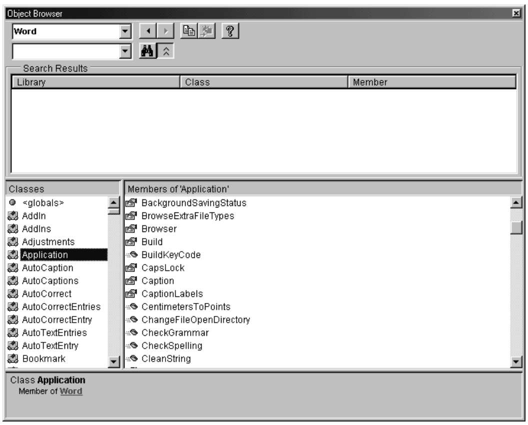
ספריית האובייקטים של Word

כדי להשתמש באובייקטי Word מתוך Visual Basic, תוכנת Word חייבת להיות מותקנת במחשב שלך. לאחר ההתקנה ספריית האובייקטים של Word אמורה להופיע בתיבת הדו-שיח Visual Basic Reference כמתואר בתרשים 22.1.



תרשים 22.1: כדי להשתמש באובייקטי Word ב- Visual Basic, הוסף ייחוס לספריית האובייקטים של Word (ייחוס Word מתוך Office 97 מוצג כאן)

כדי להוסיף ייחוס, בחר **References** מתוך תפריט **Project**, וסמן ליד Microsoft Word Object Library. לאחר יצירת הייחוס לספריית האובייקטים של Word, הקש F2 או בחר **View**, **Object Browser** כדי לראות את האובייקטים אשר Word מקצה לתוכנית כמתואר בתרשים 22.2.



תרשים 22.2: Object Browser הוא כלי יעיל לגילוי פעולות שניתן לבצע עם Word ו-Excel מתוך Visual Basic

בסעיפים הבאים תלמד כיצד לתמרן את Microsoft Word מתוך Visual Basic על ידי שימוש בספריית האובייקטים של Word.

יצירת יישומים ואובייקטי מסמכים

שני אובייקטים חשובים ביותר בתוכנת Word שיש להכיר הם: Word.Application ו-Word.Document, המאפשרים גישה מיידית ליישום Word ומסמכיו.

להפעלת Word מקוד Visual Basic, יש ליצור מופע חדש של Word.Application כפי שתעשה עם כל אובייקט אחר:

```
Dim objWord As Word.Application
Set objWord = New Word.Application
```

שים לב כי כאן משתמשים בקישור מוקדם באמצעות הצהרת Set, המקלה על תכנות באמצעות אובייקטים כמתואר בפרק 16 **מחלקות: שימוש חוזר ברכיבים**.

ראה "הצהרה ושימוש באובייקטים" בפרק 16.

שתי השורות לעיל מפעילות את Word על המחשב שלך. לאחר מכן Word מופיעה ברשימת המשימות אבל נשארת בלתי נראית עד אשר תשנה את המאפיין Visible לערך True כמתואר בדוגמה הבאה:

```
objWord.Visible = True
```

כמו כן, אפשר ליצור אובייקטים מסוג Word.Document מתוך אובייקט היישום. האובייקט Word.Application מכיל אוסף Documents. זכור מפרק 13 כי ניתן להשתמש בשיטה Add מתוך אוסף כדי להוסיף אובייקטים חדשים. כעת ביכולתך לנסות דוגמה פשוטה. תחילה צור פרויקט חדש Standard EXE והוסף ייחוס לספריית האובייקטים של Word כפי שתואר קודם לכן. לאחר מכן הוסף שני לחצני פקודה cmdStart ו-cmdClose, והכנס את הקוד מתוך תוכנית 22.1 בטופס.

תוכנית 22.1: WORDEXAMPLE.VBP - **שימוש ב-Word מתוך** Visual Basic.

```
Dim objWord As Word.Application

Private Sub cmdStart_Click()
    Dim objDoc As Word.Document

    ' Start Word and make it Visible
    Set objWord = New Word.Application
    objWord.Visible = True

    ' Create a new document
    Set objDoc = objWord.Document.Add

    ' Make it the active document
    objDoc.Activate
```

```
    ' Add some text to the Document
    objDoc.Application.ActiveWindow.Selection.InsertAfter "This is some text"
    objDoc.Application.ActiveWindow.Selection.InsertParagraphAfter
    objDoc.Application.ActiveWindow.Selection.InsertAfter "This is some more text"
    objDoc.Application.ActiveWindow.Selection.InsertParagraphAfter

    ' Make selection text bold face
    objDoc.ActiveWindow.Selection.Font.Bold = True

    ' Unselect text
    objDoc.ActiveWindow.Selection.EndOf

End Sub

Private Sub cmdClose_Click()

    ' Close Word without saving changes
    objWord.Quit False

    ' Destroy object reference
    Set objWord = Nothing

End Sub
```

כאשר אתה מפעיל את התוכנית הקודמת ולוחץ על פקד cmdStart נוצר מסמך Word חדש. בשגרת האירוע click משתמשים באובייקט selection של המסמך כדי להכניס טקסט ולהדגישו.

הערה:

בתוכנית 22.1 משתמשים במשתנה טופס לאובייקט Word.Application כדי לאפשר התייחסות מתוך מספר תת-שגרות.

בתוכנית 22.1 יצרת אובייקט מסמך מתוך אובייקט היישום. כמו כן ביכולתך ליצור אובייקט מסמך ישירות כמתואר בדוגמה הבאה:

```
Dim objDoc As Word.Document
Set objDoc = New Word.Document
ObjDoc.ActiveWindow.Selection.TypeText "Hello!"
```

הקוד הקודם יוצר מסמך Word חדש אבל בניגוד לקוד שבתוכנית 22.1 הוא משתמש במאפיין קיים של Word. במילים אחרות, אם המשתמש כבר פתח את Word, נוצר מסמך חדש ביישום הפתוח הקיים של Word. מנגד, הדוגמה הראשונה יוצרת מסמך חדש של Word בכל מקרה.

כאשר סיימת עם המסמך או אובייקט היישום, ביכולתך לסגור את המסמך על ידי שיטת Close ואת Word על ידי שימוש בשיטת Quit :

```
objDoc.close False
objWord.Quit.False
```

ערך הפרמטר האופציונלי False מורה לתוכנת Word לא לשמור שינויים. השמטת ערך זה גורמת לתוכנת Word לשאול את המשתמש במקרה והמסמך לא נשמר. בהתאם ליישום שלך ייתכן שתרצה לאפשר למשתמש לערוך את המסמך. במקרה זה פשוט השמט את השיטות quit ו-close.

שמירה, פתיחה והדפסה של מסמכים

לאחר יצירת מסמך Word ב-Visual Basic ניתן להשתמש בשיטות Save ו-SaveAs של אובייקט המסמך כדי לשמור אותו בכונן. קודם לכן יש להשתמש בשיטה SaveAs כדי לציין את שם הקובץ.

```
ObjDoc.SaveAs "C:\Temp\MyDoc.Doc"
```

לשיטה Save אין ערכים. היא נועדה לשמור מסמך לאחר שצוין שמו.

```
If objDoc.Saved = False than objDoc.Save
```

ניתן להשתמש במאפיין Saved כדי לבדוק אם המסמך נשמר מאז שבוצעו בו שינויים.

הערה:

אם תשתמש בשיטה Save ללא SaveAs אזי Word יבקש את שם הקובץ.

כדי לפתוח מסמך קיים, צור שיגרה של אובייקט Word.Application, והשתמש בהצהרה Set ביחד עם Open מתוך האוסף Document.

```
Set objDoc = objWord.Documents.Open ("C:\Temp\Junk.Doc")
```

בשורת הקוד הקודמת, אם הקובץ Junk.Doc לא קיים, Word תציג הודעת שגיאה. ניתן לטפל במצב זה מתוך Visual Basic על ידי שימוש בהצהרה OnError, ובדיקת אובייקט המסמך כדי לראות האם המסמך פתוח.

```
On Error Resume next
Set objDoc = objWord.Documents.Open ("C:\Temp\Junck.Doc")
If objDoc Is Nothing Then MsgBox("Open was not successful")
```

כדי להדפיס מתוך Word השתמש בשיטה PrintOut ו-PrintPreview. השורה הבאה גורמת לתוכנית Word להדפיס מסמך :

```
objDoc.PrintOut
```

אם אתה יוצר דוח ב-Word ייתכן שתרצה להשתמש בשיטה PrintPreview כדי לאפשר למשתמש לראות את המסמך ולהחליט האם להדפיסו.

עבודה עם טקסט

בתוכנית 22.1 השתמשת בשיטות InsertAfter, InsertParagraphAfter כדי להוסיף טקסט למסמך Word. אולם קיימות עוד דרכים רבות לעבוד עם טקסט בתוך Word.

לדוגמה, אפשר להשתמש באוסף Words כדי לאחזר מילה אחת בכל פעם. הדוגמה הבאה מוסיפה את המילים לאובייקט Word.Document בתיבת רשימה.

```
For nWord = 1 to objDoc.Words.Count
    lstWords.AddItem "Word "& "is " & objDoc.Words(nWord).Text
Next nWord
```

כמו-כן ניתן להחליף מילים בתוך מסמך על ידי הצבת ערך למאפיין Text אבל יש לזכור כי החלפת מילה במספר מילים תשנה את סך המילים במסמך.

הערה:

האובייקט Word.Document מכיל גם אוספים של Sentences (משפטים) ו-Paragraphs (פסקאות).

דרך קלה להכניס טקסט במיקום מסוים היא על ידי שימוש בסימניות. בתוכנת Microsoft Word ניתן למקם את הסמן במיקום ספציפי ולבחור Bookmark מתוך תפריט Insert כדי ליצור סימניות. מתוך הקוד ניתן להשתמש באוסף Bookmarks כדי לזהות את הסימניות.

```
Set objDoc = objWord.Documents.Add  (D:\DATA\MyTemplate.DOT")
ObjDoc.Bookmarks ("Name").Range.Text = "John Smith"
ObjDoc.Bookmarks ("Address").Range.Text = "123 Fourth Street"
ObjDoc.SaveAs "D:\DATA\JSMITH.DOC"
```

בדוגמה זו יצרת מסמך חדש מתוך תבנית לקוח, הוספת נתונים ואת מיקום הסימניות, ושמרת את המסמך לקובץ חדש. שים לב כי הקוד מניח כי תבנית המסמך MyTemplate מכילה סימניות מוגדרות מראש.

תכונות שימושיות נוספות

אם אתה רוצה ליצור דוח באמצעות Word, עליך לקרוא לשיטות ולהגדיר תכונות שתשלוטנה על יצירת הדוח. דרך אחת ליצור דוח שייראה מסודר היא על ידי הוספת טבלאות לאוסף Tables כמתואר בקוד הבא:

```
Dim objTable As Word.Table
Set objTable = objDoc.Tables.Add (objDoc.Range, 10, 2)
objTable.Cell(1, 1).Range.Text = "Hello"
objTable.Cell(1,2).Range.Text = "Dolly"
objTable.Columns(2).AutoFit
```

הקוד הקודם מכניס טבלה של עשר שורות ושתי עמודות במסמך ומוסיף טקסט. לבסוף, השיטה AutoFit מופעלת כדי להתאים את רוחב העמודות לרוחב הטקסט בעמודה.

מאפיין מעניין נוסף הוא היכולת להוסיף משתנים משלך למסמך פתוח. מאפיין זה שימושי למשל אם קיימים מספר מסמכים פתוחים ויש צורך לזהותם. כדי להוסיף משתנה, צור שם והשתמש בשיטה Add באוסף Variables.

```
ObjDoc.Variables.Add "EMPLOYEEID", "8675309"
```

תוכל להשתמש במשתנה שלך כדי לזהות את המסמך

```
For Each objDocTemp In ObjWord.Documents
	If objDocTemp.Variables ("EMPLOYEEID") = "8675309" Then
		Set objDoc = objDocTemp
	End If
Next objDocTemp
```

כדי שהקוד שתואר יעבוד, עליך להציב את הערך של objDoc ל-Nothing אם השתמשת בו קודם.

Word.Basic

בסעיפים הקודמים למדת כיצד להשתמש בספריית האובייקטים של Microsoft Word. השתמשת בקישור מוקדם, פירוש הדבר שמשתנים הוכרזו תחילה כסוג מסוים ולא כאובייקט מסוים כללי יותר. אולם אם שמרת יישומים ישנים יותר של Visual Basic ייתכן כי תמצא ספריה שונה - ספריית WordBasic. יישומים המשתמשים באובייקט זה, מצהירים בדרך כלל עליו כקשר מאוחר כפי שמופיע בדוגמה הבאה:

```
Dim objWord As Object
Set objWord = CreateObject ("Word.Basic")
```

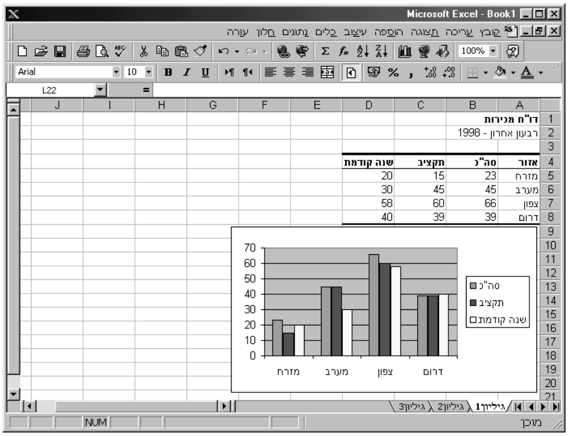
גם הפקודות שונות במקצת:

```
objWord.FileNew
objWord.StartOfDocument
objWord.FontSize 24
objWord.Insert "Some Big Text"
objWord.FontSize 12
objWord.InsertPara
objWord.Insert "Some Smaller Text"
```

שימוש בקוד לפי הדוגמה הקודמת בהחלט אינה שיטה רצויה לגשת ל-Word מ- Visual Basic, אולם ציינו זאת כדי למנוע בלבול במקרה ותיתקל בקוד ישן יותר.

עבודה עם Excel

באמצעות הטכניקות לשליטה ב-Microsoft Word ניתן לשלוט גם ב-Excel מתוך יישומי Visual Basic. כמו Word גם ל-Excel יש ספריית אובייקטים משלה אשר יש לייחסה בתוך פרויקט Visual Basic. על ידי שימוש באוטומציית OLE עם Excel אפשר ליצור דוחות מסודרים כמו זה המתואר בתרשים 22.3. העימוד והעיצוב של הדוח נעשה בתוכנת Excel ובאמצעות Visual Basic הוזנו הנתונים.



תרשים 22.3: דוח זה המכיל גרף ועמודות מספרים קשה מאוד ליצירה באמצעות פקודות ההדפסה המובנות ב- Visual Basic

יצירת אובייקטי Excel

כדי להפעיל תבנית Excel השתמש באובייקטים Workbook ו-Worksheet. הקוד הבא מפעיל את Excel ויוצר חוברת עבודה חדשה המכילה גיליון אחד.

```
Dim objExcel As Excel.Application
Set objExcel = New Excel.Application
ObjExcel.Visible = True
ObjExcel.SheetsInNewWorkbook =1
ObjExcel.Workbooks.Add
```

הצבת ערכים של תאים וטווחים

כדי להציב ערך ייחודי של תא בגיליון אפשר להשתמש באוסף Cells מהמאפיין Range של האובייקט Worksheet :

```
With objExcel.ActiveSheet
.Cells (1,2).Value = "10"
.Cells (2,2).Value = "20"
.Cells (3,2).Value = "Sum (B1:B2)"
.Range ("A3") = "Total"
End With
```

הערה:

Range עובד גם עם תחום שמות של תא אחד או יותר.

```
ObjExcel.ActiveSheet.Range("MyRange").Font.Color = vbRed
```

שים לב כי השתמשנו בהצהרת With כדי לא לחזור על שם האובייקט. ב-Excel ייתכן שתמצא כי האובייקט שאליו עליך לפנות, נמצא מספר שלבים עמוק באוסף.

```
ObjExcel.Workbooks(1).Sheet("Portfolio").Range(2,4).Font.Bold = True
```

למרות שגישה זו עלולה להיות מבלבלת היא דרך טובה לארגן מספר גדול של מחלקות. כאשר אתה מפתח ב-Excel וב-Word אפשר להשתמש במספר שיטות כדי להפעיל את האוסף בצורה קלה יותר :

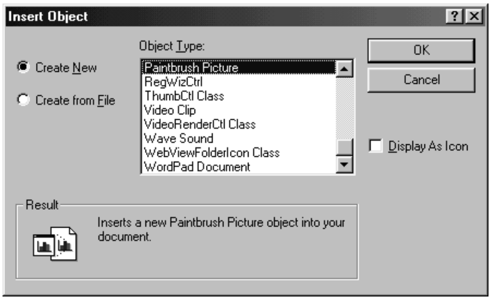
- צור אובייקט משתנה ביניים.
- השתמש בתו נקודה (Dot Notation) כפי שתואר בשורת הקוד האחרונה.
- השתמש בהצהרת With כדי להימנע מהקלדת יתר.

שימוש בפקד המכולה OLE

פקד המכולה OLE מאפשר לך להטמיע או לקשר בין אובייקטי OLE ביישום שלך. בסעיפים הקודמים למדת כיצד להשתמש באובייקט OLE לשלוט על יישום אחר. שימוש בפקד המכולה OLE שונה בכך שאובייקט חיצוני מופיע ביישום שלך כטופס ולא כיישום נפרד. במילים אחרות, אפשר ליצור טופס עם פקדים, תוויות ופקד OLE המכיל מסמך Word. אפשר ליצור את האובייקט המוטבע בזמן העיצוב או בזמן ההפעלה, כל עוד פקד OLE הוכנס לטופס.

יצירת אובייקט מוטבע בזמן העיצוב

כדוגמה, צור אובייקט מוטבע של תוכנת הציור Paintbrush של Windows. תחילה הפעל פרויקט חדש מסוג Standard EXE וגרור את הפקד המכיל OLE לטופס. ברגע שגררת את הפקד, תיבת הדו-שיח InsertObject תופיע כמתואר בתרשים 22.4.



תרשים 22.4: InsertObject מאפשרת לציין את אובייקט יוטבע בזמן העיצוב

תיבת הדו-שיח InsertObject מכילה רשימה של כל האובייקטים שניתן להשתמש בהם באמצעות פקד OLE. אם לא תבחר אובייקט, פקד OLE ייצור מסגרת לאובייקט שניתן ליצור בזמן ההפעלה. לדוגמה, בחר בתוכנת Paintbrush ולחץ על OK.

מצב אובייקט

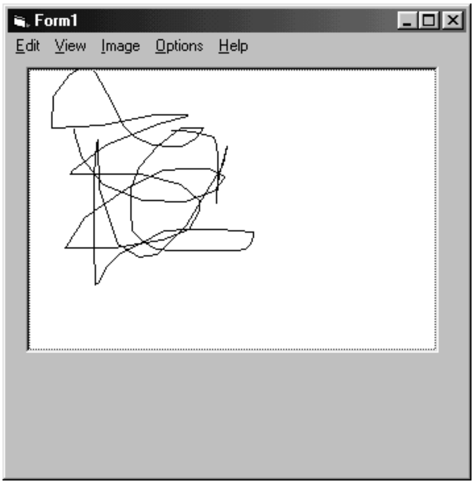
לאחר שבחרת בסוג אובייקט מסוים ולחצת על OK, סרגל הכלים והתפריטים של Visual Basic משתנים. שינוי זה התרחש כיון שאובייקט OLE המוכל בפקד הופעל. כאשר האובייקט מופעל אפשר לתקשר איתו. בפרויקט לדוגמה, כיון שיצרת אובייקט מסוג תמונה של Paintbrush, אתה יכול לצייר בתוך המסגרת ולבצע את כל הפעולות כמו בהפעלה רגילה של התוכנה. צייר מספר קווים ונטרל את פעולת האובייקט על ידי לחיצה מחוץ לפקד שעל הטופס. שים לב כי עדיין תוכל לראות את הציור במסגרת האובייקט הלא פעיל אבל הסביבה של Visual Basic חזרה למצב העבודה הרגיל. כאשר אובייקט אינו פעיל אפשר לתקשר איתו. המאפיין DisplayProperty של פקד OLE מגדיר האם האובייקט יוצגו כסמל או יוצג תוכנו (במקרה זה הציור).

הפעל את התוכנית על ידי הקשת F5 ושים לב כי התנהגות האובייקט נותרה ללא שינוי. לחיצה כפולה על האובייקט מפעילה את Paintbrush כך שניתן לערוך את האובייקט כמתואר בתרשים 22.5.

שמירה והטבעה של אובייקט לקובץ

כעת תוכל להוסיף יכולת של שמירה וטעינה לאותו יישום. הוסף שני לחצני פקודה cmdSave ו-cmdLoad לטופס והכנס את הקוד הבא:

```
Private Sub cmdSave_click()
    Open "C:\OBJECT.DAT" For Binary As #1
    OLE1.SaveToFile 1
    Close #1
End Sub
Private Sub cmdLoad_Click ()
    Open "C:\OBJECT.DAT" For Binary as #1
    OLE1.ReadFromFile 1
    Close #1
End Sub
```



תרשים 22.5: שים לב כי פקודת התפריט File של אובייקט פעיל לא מופיעה בצורה אוטומטית כיון ששמירת הקובץ נתונה לשיקול דעתו של המפתח

שים לב כי על אף השימוש בשיטת פקד OLE לשמירה וטעינה של קובץ, עדיין מוטלת עליך פעולת טעינת הקובץ באמצעות המשפט Open. אפשר לבחון את הקוד על ידי שמירת הציור, שינויו במקצת וטעינה מחודשת של הקובץ לתוך הפקד.

יצירת אובייקט מוטבע בזמן ריצה

בדוגמה הקודמת, יצרת אובייקט מוטבע של Paintbrush בזמן עיצוב. אפשר גם ליצור פקד OLE ריק בזמן העיצוב ולטעון אובייקט בזמן ההפעלה.

כדי ליצור פקד OLE ריק, פשוט בחר Cancel כאשר תיבת הדו-שיח InsertObject מופיעה או לחץ לחיצה ימנית על הפקד ובחר Delete Embedded Object אם אחד כזה כבר קיים.

צור אובייקט מוטבע בזמן ההרצה על ידי שימוש בשיטת פקד OLE: CreateEmbed. שיטה זו דורשת שני ערכים: מסמך מקור ושם המחלקה של האובייקט. את ערך שם אובייקט המקור ניתן להשאיר ריק אם מעונינים ליצור אובייקט ריק.

שורת הקוד הבאה יוצרת תמונה של Paintbrush בפקד OLE בשם OLE1:

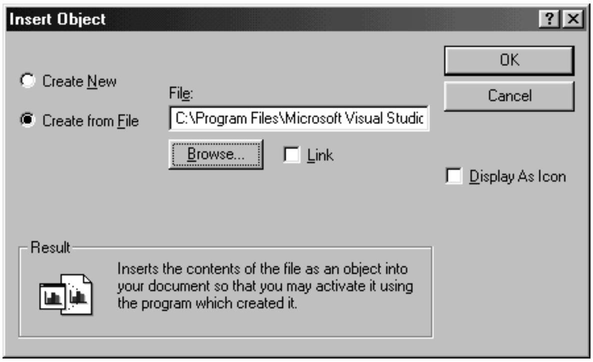
```
OLE1.CreateEmbed " ", "Paint.Picture"
```

שים לב כי עליך לציין את שם המחלקה כדי ליצור את האובייקט. ניתן לראות רשימת מחלקות תקפות על ידי לחיצה על הסוגריים (...) בחלון המאפיינים של Class Property שבפקד OLE.

יצירת אובייקט מקושר

פקד OLE גם מסוגל ליצור אובייקטים מקושרים. אובייקט שכזה, מקושר לקובץ חיצוני ולא מוטבע בקובץ עצמו. כאשר מפעילים אובייקט מקושר, חלון היישום מופיע כדי שנוכל לערוך את האובייקט.

ליצירת מסמך מקושר, עליך לציין את שם מסמך המקור. עבור אובייקט מסוג תמונה של Paintbrush עליך לציין את מיקומו של קובץ מסוג תמונת מפת סיביות (bitmap) כמתואר בתרשים 22.6.



תרשים 22.6: ליצירת אובייקט מקושר בזמן העיצוב בחר בשם הקובץ וסמן את תיבת הבחירה Link

כמו כן תוכל ליצור אובייקט מקושר בזמן ההרצה על ידי שימוש בשיטה CreateLink כמתואר בדוגמה הבאה:

```
OLE1.CreateLink "C:\Windows\Pinstripe.BMP"
```

כאשר הינך משתמש באובייקט מקושר, שינוי קובץ האובייקט ישתקף ביישום שלך. כמו כן, אם תעדכן אובייקט המקושר ליישום, השינויים יתבצעו גם בקובץ המקור. האופן בו האובייקט מתעדכן מבוקר באמצעות המאפיין UpdateOption היכול לקבל את הערכים הבאים:

שם משתנה	תאור
vbOLEAutomatic	האובייקט מעודכן אוטומטית כאשר משנים את קובץ המקור (ברירת מחדל)
vbOLEFrozen	האובייקט מעודכן כאשר קובץ המקור נשמר מיישום המקור
vbOLEManual	האובייקט מעודכן כאשר מתבצעת קריאה לשיטה Update

בדוגמה שבקטע הקודם, שינוי המאפיין UpdateOptions מגדיר מתי ישתקפו בפקד OLE השינויים שבוצעו בתמונה.

מכאן...

בפרק זה למדת להשתמש בפקד OLE לשליטה על יישומים אחרים ולהטביע אובייקטים בתוכנית. למרות שהתמקדנו ביישומים מסוימים, זכור כי ייתכן שברשותך יישומים שלא פותחו על ידי Microsoft אבל ספריית הסוגים שלהם עשויה להיות מותקנת במערכת שלך. הרחבה של החומר הנדון בפרק זה תמצא בפרקים הבאים:

- ללימוד נוסף על מחלקות ואוספים ראה פרק 16: **מחלקות: שימוש חוזר ברכיבים**.
- להבין כיצד תוכל לכלול נתונים בדוחות שלך ראה פרק 24 **יסודות מסד הנתונים**.

23

ארגז הכלים של המומחה

מה בפרק זה?

- שיחה מזוהה
- בניית שומר מסך
- תוכנית ייצוא טבלאות משרת SQL למסד נתונים של Access
- שימוש ב-API של Windows ליצירת תמונות שקופות

פרק זה נועד לאנשים שלומדים מדוגמאות. בעקבות המהדורה הקודמת, רבים מהקוראים ביקשו יותר דוגמאות. לאור פנייתם הוכנסו לפרק זה מספר רב של פרויקטים בנויים באמצעות Visual Basic. בנוסף, הפרויקטים בפרק זה מאחדים יישומים מעשיים מפרקים אחרים. לאחר שתתנסו מעט בעבודה עם Visual Basic, נסו חלק מהם - רובם מהנים מאוד.

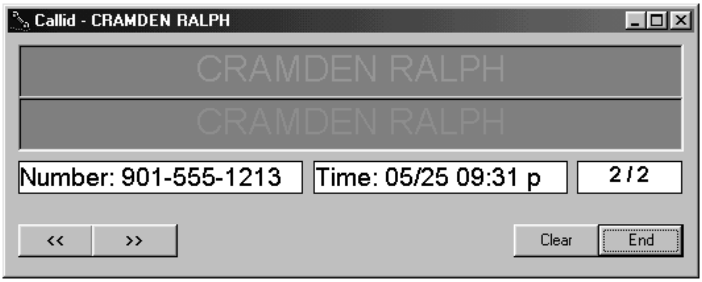
שיחה מזוהה

שיחה מזוהה (Caller ID) הינו שירות הניתן על ידי חברות הטלפונים ומאפשר לך לראות את שם המטלפן או מספרו. המידע מועבר בין הצלצולים כך שתוכל לראות את פרטי המטלפן לפני הרמת השפופרת.

בעידן זה של שיחות טלמרקטינג לא מוזמנות, שיחה מזוהה היא המצאה גדולה, אבל, יש גם מלכוד. כדי לסנן שיחות, עליך להימצא ליד הצג כאשר הטלפון מצלצל - אלא אם תכתוב יישום Visual Basic כמתואר בסעיף זה (ראה תרשים 23.1). יישום זה מדמה תצוגת שיחה מזוהה בתוספת אחת - צליל.

כאשר הטלפון מצלצל, היישום משמיע צליל שמוצמד למספר מסוים. לדוגמה, תוכל לגרום לכרטיס הקול שלך לצעוק "אל תענה!!" כאשר אנשי טלמרקטינג עושים את הסבב שלהם.

למרות שיישום זה מאוד ממוקד, בזמן בנייתו תלמד כיצד לתקשר עם המודם מתוך Visual Basic, טכניקה שניתן להשליך גם לתחומים אחרים.



תרשים 23.1: היישום "שיחה מזוהה" מתריע על ידי הפעלת קובץ שמע

דרישות לשימוש בקובץ לדוגמה

היות ותוכנה זו מתקשרת עם "העולם האמיתי", יש למלא אחר מספר דרישות בסיסיות כדי להשתמש בה.

- עליך להירשם לשירות שיחה מזוהה.
- ברשותך צריך להיות מודם מחובר לקו טלפון, ותומך בשירות שיחה מזוהה.
- עליך לדעת את הפקודות שיגרמו למודם להציג את המידע.

לפני שתכתוב קוד Visual Basic, יש לזהות את הפקודות הדרושות לשימוש בשיחה מזוהה ולבדוק אותן בתוכנית מסוף כמו HyperTerminal. לדוגמה, הקלדת הפקודה AT#CID=1 תגרום למודם שלי להפעיל את תצוגת השיחה המזוהה ולאחר שהטלפון יצלצל יציג את המידע בהפעלת (Session) המסוף כמתואר:

```
RING
DATE = 0417
TIME = 2005
NMBR = 9015551212
NAME = SILER BRIAN
RING
```

הערה:

התצורה וההגדרה של שיחה מזוהה עשויות להשתנות ממודם למודם. ליתר דיוק מודם ISDN מחזיר את המידע באופן שונה לגמרי. תוכל לשנות בקלות את היישום ולהתאימו למודם שלך, אך היכרות עם תצורת השיחה המזוהה שלו יכולה להועיל.

לאחר שבדקת כיצד לקבל את המידע על השיחה המזוהה מהמודם, בעזרת תוכנית עזר, השלב הבא הוא כתיבת יישום Visual Basic המדבר עם המודם, מקבל מידע ופועל בהתאם. היישום המודגם בסעיף זה, מבצע פקודת הפעלה לתצוגת השיחה המזוהה ולאחר מכן גורם למודם לבדוק בצורה מחזורית מידע על זהות המטלפן. אם מידע כלשהו מופיע, היישום ישמיע קובץ wav (קול) מתאים. לב התוכנית הוא הפקד MSCOMM שהוא המרכיב המאפשר לתקשר עם המודם מתוך Visual Basic.

טכניקות Visual Basic בהן נשתמש

בנוסף לסקירה הדנה בפקד MSCOMM, התוכנית מצרפת נושאים שכוסו בפרקים אחרים של הספר. קבצי INI, פקד Timer, קריאות API. כדי להשלים דוגמה זו בהצלחה, עליך להבין היטב כל אחד מרעיונות אלה.

השמעת צליל

שימוש בקריאת API נועד להשמיע צליל באמצעות כרטיס הקול. דוגמה ספציפית זו מכוסה בפרק 20: **גישה ל-API של Windows**. גש לפרק זה אם הינך צריך עזרה בשימוש בקריאת API, sndPlaySound.

ראה "קריאות API שימושיות" בפרק 20.

שימוש בפרקי זמן (Intervals)

פקד Timer המכוסה בפרק 4: **שימוש בפקדי ברירת המחדל של Visual Basic** משמש לעדכון תהליך בדיקת המידע עבור שיחה מזוהה. בתוכנית הדוגמה, האירוע Timer מקודד בשורה בודדה כמתואר:

```
Private Sub tmrMain_Timer()
    ChekForCall
End Sub
```

ChekForCall היא פונקציה מותאמת לבדיקת מידע חדש המתקבל מהמודם.

ראה "פקדים למטרות מיוחדות" בפרק 4.

קובץ קונפיגורצית INI

בקובץ INI מידע על כל צלצול וגם על הנתיב בו נמצא כל קובץ wav שתרצה להשמיע למטלפן מסוים. כמובן שתוכל לשמור נתונים אלה במאגר נתונים או קובץ אחר, אבל בחרתי בקובץ INI כיון שאינו בזבזני וקל לעריכה. דוגמה לקובץ INI מתוארת כאן:

```
[General]
ComPort = 3
InitString="AT#CID=1"
CallCount=3

[Xref]
VANDELAY INDUST=Dad at Work
PIERCE JAMES J=Nelda and Jerry Pierce
TAYLOR TECHNOLOG = This is a sales call, don't answer !
MEMPHIS, TN=Cellular phone in Memphis
[Sounds]
PAY PHONE=d:\wav\Dad.wav
VANDELAY INDUST=d:\wav\Hello.Wav

[Call1]
Time=05/25 03:05 p
Number=901-362-6030
Name=LAZENBY N D

[Call2]
Time=05/25 09:31 p
Number=901-555-1212
Name=PRENTICE BRUCE

[Call3]
Time=05/26 12:43 p
Number=901-754-7222
Name=FRIDAY JOE
```

קובץ INI מחולק לתת-סעיפים :

- [Sounds] - סעיף זה מכיל את הנתיב לקבצי wav המתאימים לשם כל מטלפן.
- [XREF] - סעיף זה מכיל כינויים עבור שם כל מטלפן.
- [CALL##] - התוכנית יוצרת מקטע חדש לכל שיחה שהתקבלה.
- [GENERAL] - סעיף זה מכיל את המספר העדכני של השיחות [CALL] שהתקבלו ואת פרטי האתחול.

התוכנית משתמשת בפונקציות sGetINIString ו-writeINIString המתוארות בפרק 21 : **עבודה עם קבצים** לשליטה על קבצי INI.

ראה "הבנת קבצי INI" בפרק 21.

התחלת התוכנית

הסעיף הבא יתמקד בחומר חדש הקשור לפקד התקשורת. השלם את הצעדים הבאים :

1. התחל פרויקט חדש מסוג Standard EXE.
2. הוסף מודול לפרויקט.
3. הוסף את הקוד המופיע בפרק 21, הנחיות לגישה לקבצי INI.
4. הוסף קוד לקריאות API עבור sndPlaySound מפרק 20.
5. גרור פקד TIMER לטופס בשם trtMain וקבע את מרווח הזמן ל-900.
6. הפוך את SubMain לאובייקט Startup של הפרויקט.
7. התחל ביצירת קובץ INI, בהתבסס על הדוגמה מהסעיף הקודם.

הגדרת פקד התקשורת (Communications Control)

הפקד MSComm מאפשר ל- Visual Basic לשלוח ולקבל מידע דרך יציאת תקשורת טורית או מודם המוכר כיציאת Com, בדומה מאוד לתוכנת ההדמיה של המסוף. לפני שתוכל להשתמש בפקד, עליך להוסיפו לארגז הכלים. כדי לבצע זאת, לחץ על הלחצן הימני באזור ריק של ארגז הכלים ובחר Components מתוך התפריט. מתיבת הדו-שיח Components סמן ליד Microsoft Comm Control 6.0 ולחץ OK. הפקד אמור להופיע בארגז הכלים. לאחר מכן גרור את הפקד לטופס. שים לב כי הוא תמיד מופיע כסמל ולא משנה כמה גדול תנסה לציירו. כדי להתאים את הפקד MSComm לשימוש במודם, עליך להגדיר את המאפיינים.

- CommPort - מספר שלם המייצג את יציאת COM אליה מחובר המודם. למשל 2 עבור COM2.
- Settings - מחרוזת המגדירה את קצב התקשורת והגדרת הזוגיות (Parity).

שני מאפיינים אלה ניתן להגדיר בכל זמן, בתחילת התוכנית באירוע Load או בשיגרה SubMain. אם המודם מחובר אל Com3 ההצהרות תיראנה כך:

```
Form1.MSComm1.CommPort = 3
Form1.MSComm1.Settings = "9600,N,8,1"
```

גם אם תגדיר את המאפיינים בזמן עיצוב התוכנית, עדיין יהיה עליך לבצע מספר פעולות. בתחילת התוכנית: אתחול פקד התקשורת ושליחת פקודות למודם להפעלת תצוגת השיחה המזוהה. השיגרה המלאה SubMain מוצגת בתוכנית 23.1.

תוכנית 23.1: CALLID.VBP שגרת אתחול לתוכנית שיחה מזוהה

```
Sub Main()
    Dim nComPort As Integer
    Dim sInit As String
    Dim sTemp As String
    Dim bStop As Boolean

    sINIfile = App.Path & "\CALLID.INI"
    nComPort = CInt(sGetINIString(sINIfile, "General", "ComPort", "2"))
    sInit = sGetINIString(sINIfile, "General", "InitString", "AT#CID=1")
    With frmMain
        ' SET UP THE COM PORT
        .MSComm1.CommPort = nComPort
        .MSComm1.Settings = "9600,N,8,1"
        .MSComm1.InputLen = 0

        ' OPEN THE CONNECTION AND SEND THE INIT STRING
        .MSComm1.PortOpen = True
        .MSComm1.Output = sInit + Chr$(13)

        ' WAIT A FEW SECONDS FOR MODEM RESPONSE
        nTemp = 0
        bStop = False
        Do While nTemp < 32000 And bStop = False
            nTemp = nTemp + 1
            If .MSComm1.InBufferCount >= 2 Then
                sTemp =.MSComm1.Input
                If InStr(sTemp, sInit) = 0 Then bStop = True
            End If
        Loop

        ' IF THE MODEM DIDN'T SAY OK THEN END
        If InStr(sTemp, "OK") = 0 Then
            MsgBox "Modem did not respond with OK.", vbOK + vbCritical, _
                "Initialize Error"
            End
        End If
```

```
            DoEvents

            ' CLEAR REMAINING INPUT
            sTemp = .MSComm1.Input

            ' START THE POLLING TIMER
            .tmrMain.Enabled = True

      End With

      ' DISPLAY CALLER INFORMATION FROM LAST TIME
      nCount = CInt(sGetINIString(sINIfile, "General", "CallCount", "0”))
      DisplayList nCount
      frmMain.Show

End Sub
```

הקוד המתואר בתוכנית 23.1 מבצע רק את אתחול התוכנית ומדגים כיצד לשלוח ולקבל פקודות מהמודם. הפקודות נשלחות למודם באמצעות המאפיין Output של הפקד MS Comm (שים לב כי מתוסף Carriage Return בדיוק כאילו הקלדת אותו). במילים אחרות, אם תרצה לחייג מספר באמצעות פקודות ATDT, פשוט הוסף את הפקודה למאפיין Output. בתוכנית 23.1 מנצלים מאפיין זה כדי לשלוח מחרוזת עדכון מקובץ INI למודם.

אם תיזכר בתהליך שליחת פקודות למודם, תיווכח כי המודם מגיב בהודעת מצב כמו OK, הודעת מידע כמו: זהות המתקשר, או הגדרת Register. פקד MS Comm לוכד הודעות אלו ושומר אותן במאגר מסוים. הינך משתמש במאפיין Input של הפקד כדי למשוך את המידע מהמאגר לתוכנית בדרך כלל על ידי הכנסתו למשתנה. בתוכנית 23.1 בודקים את המאפיין InputBuffer כדי לגלות אם קיים מידע המחכה במאגר. אם אכן קיים מידע כזה, תוכל למשוך אותו באמצעות הפונקציה Instr ולבדוק האם המודם הגיב למחרוזת האתחול.

הערה:

המאפיין InputLen שולט על מספר התווים הנמסרים על ידי המאגר כאשר ניגשים למאפיין. הצבת הערך 0 כפי שנעשה כאן גורמת לכל תא במחרוזת להיות מוחזר.

אם המודם מגיב OK, הצלחת לתקשר איתו והינך יכול להתחיל בתהליך בדיקת המידע של השיחה המזוהה. כדי לעשות כן עליך לשנות את המאפיין Enable של פקד Timer לערך True, ובכך להתחיל את התהליך. הנושא האחרון שנשאר בפונקציות Main הוא הצגת פרטי המתקשר האחרון (השמור בקובץ INI בטופס). השיגרה DisplayList פשוט קוראת את הערכים מתוך [CALL##] בקובץ INI, ומשנה את התוויות בטופס.

בדיקת השיחות

השגרות החשובות בדוגמת השיחה המזוהה הן CheckForCall ו-WriteCIDData המוצגות בתוכנית 23.2. CheckForCall מטפלת במידע המתקבל מהמודם בעוד WriteCIDData מאחסנת את פרטי השיחה המזוהה לקובץ INI.

תוכנית 23.2: CALLID.VBP שיגרה המתרגמת את הנתונים מ- MS Comm Control.

```
Sub CheckForCall()
	Dim nPrevCount As Integer
	Dim l As Long
	Dim sFile As String

	With frmMain
		If .MSComm1.InBufferCount >= 2 Then

			' STOP TIMER AND GET INPUT FROM COM PORT
			.tmrMain.Enabled = False
			sTemp = .MSComm1.Input
			' If input is a small string then ignore,
			' It might just be the first 'RING' signal
			If Len(sTemp) < 10 Then
				.tmrMain.Enabled = True
				Exit Sub
			End If

			' STORE CALL COUNTER, ADD INFO TO INI FILE
			nPrevCount = nCount
			WriteCIDData (sTemp)

			' IF NO DATA WAS FOUND THEN EXIT
			If nCount = nPrevCount Then
				ClearStuff
				.lbldbName = "No Data Sent " & Now
				Exit Sub
			End If

			' DISPLAY CALLER INFORMATION ON THE FORM,
			' PLAY THE SOUND FILE A COUPLE OF TIMES
			DisplayList nCount

			sTemp = sGetINIString(sINIfile, "Call" & nCount, "Name", "Unknown")
			sFile = sGetINIString(sINIfile, "Sounds", sTemp, "?")
			If sFile <> "?" Then
				l = sndPlaySound(sFile, SND_SYNC)
				l = sndPlaySound(sFile, SND_ASYNC)
			End If
		End If
```

```
            ' CLEAR ANY EXTRA INPUT FROM THE COMM CONTROL
            sTemp = .MSComm1.Input

            ' RESTART TIMER FOR NEXT CALL
            .tmrMain.Enabled = True
      End With
End Sub

Sub WriteCIDData(sInput As String)
      Dim sName As String
      Dim sNumber As String
      Dim sTime As String
      Dim sDate As String
      Dim sSection As String

      'PARSE EACH PIECE OF INFORMATION FROM THE INPUT STRING
      'THIS WILL VARY FROM MODEM TO MODEM (MINE IS A CARDINAL 33.6)

      sNumber = "?"

      If InStr(sInput, "MESG =") Then
            nTemp = InStr(sInput, "MESG =")
            sName = Mid(sInput, nTemp + 7)
            sNumber = "NO NUMBER SENT"
      Else
            nTemp = InStr(sInput, "NMBR =")
            If nTemp <> 0 Then sNumber = Mid(sInput, nTemp + 7, 10)

            nTemp = InStr(sInput, "NAME =")
            If nTemp <> 0 Then sName = Mid(sInput, nTemp + 7)
            nTemp = InStr(sInput, "DATE =")
            If nTemp <> 0 Then sDate = Mid(sInput, nTemp + 7, 4)
            nTemp = InStr(sInput, "TIME =")
            If nTemp <> 0 Then sTime = Mid(sInput, nTemp + 7, 4)
            sTemp = Left$(sNumber, 3) & "-" & Mid$(sNumber, 4, 3) & "-" & _
                  Right$(sNumber, 4)
            sNumber = sTemp
      End If
      If sNumber = "?" Then Exit Sub
      'WRITE INFORMATION TO THE INI FILE
      nCount = nCount + 1
      writeINIString sINIfile, "General", "CallCount", CStr(nCount)
      sSection = "Call" & nCount
      WriteINIString sINIfile, sSection, "Time", Format$(Now, "mm/dd hh:mm a/p")
      WriteINIString sINIfile, sSection, "Number", sNumber
      WriteINIString sINIfile, sSection, "Name", sName
End Sub
```

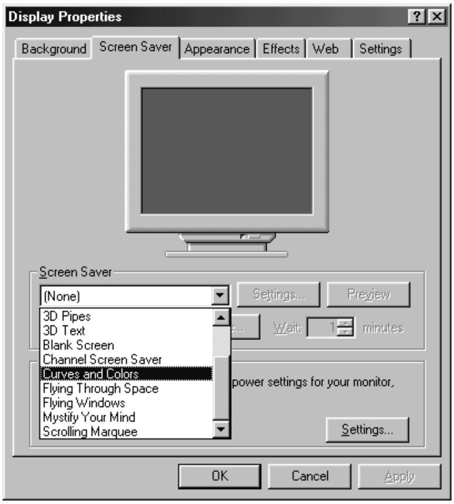
הסיבה שנחוצות הרבה קריאות Instr נובעת מכך שבזמן קבלת המידע באמצעות פקד MS Comm, הוא מגיע כמחרוזת אחת עם Carriage Return ו- Line Feed בתוכה.

שים לב כי WriteCIDData שולחת מידע לקובץ INI והמידע נקרא בחזרה בשדות נפרדים, על ידי פונקציית CheckForCall.

בניית שומר מסך באמצעות Visual Basic

Windows מגיעה עם תוכנית שומר מסך המופעלת בצורה אוטומטית לאחר פרק זמן של חוסר פעילות. הרעיון העומד מאחורי שומר המסך היא שהמסך "יישמר" מקבלת מידע קבוע ו"צריבת" נתונים. צריבה זו היתה בעיה נקודתית במסופי טקסט בלבד שעליהם ישב תפריט קבוע למשך שעות. מאותה סיבה קיימת אזהרה בספרי ההדרכה של טלוויזיות בפני שימוש במשחקי וידאו.

ניתן להגדיר שומר מסך מתוך הכרטיסיה Screen Saver של תיבת הדו שיח Display Properties (מאפיינים תצוגה), כמתואר בתרשים 23.2.



תרשים 23.2: תוכל לכתוב תוכנית Visual Basic המתנהגת כשומר מסך

שומרי מסך הפכו לאומנות ממש: מכתובית מטיילת ועד לאובייקטים תלת-מימדיים שבתוכם אפשר להציג תמונה נבחרת. אולם, אם תרצה להיות באמת יצירתי תוכל ליצור שומר מסך משלך. ב- Visual Basic משימה זו פשוטה מאוד. שומר המסך נכתב בדיוק כתוכנית רגילה אבל תכנון האירועים נעשה בצורה שונה. לדוגמה, תזוזת עכבר תפסיק את התוכנית. בסעיף הבאה, ניצור שומר מסך פשוט.

הגדרת הטופס הראשי (MainForm)

התחל פרויקט Standrd.EXE ושנה את מאפייני הטפסים כדלקמן :

1. קבע את מאפיין WindowState לערך Maximized.
2. קבע את BorderStyle לערך None.
3. קבע את BackColor לצבע שחור.

כידוע, שומר המסך מפסיק את פעולתו כאשר המשתמש מזיז את העכבר או מקיש על מקש כלשהו. הוסף הצהרת End לאירועי MouseMove ו-KeyPres בטופס.

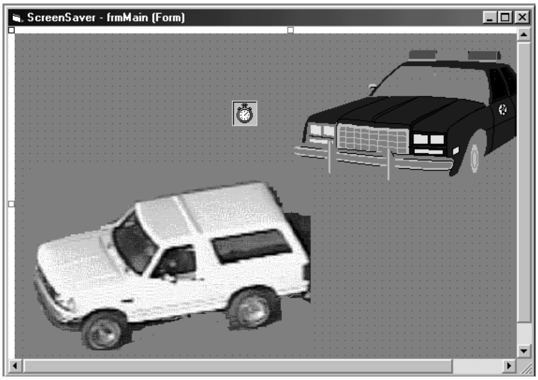
```
Dim nMouseCount As Integer
Private Sub Form_KeyPress (KeyAscii as Integer)
	End
End Sub

Private Sub Form_MouseMove (Button As Integer, Shift As Integer, X As Single, _
		Y As Single)
	nMouseCount = nMouseCount + 1
	If nMouseCount > 5 Then End
End Sub
```

שים לב כי משתמשים במונה כך שהתוכנית אינה מפסיקה את פעולתה באירוע MouseMove הראשון. הינך זקוק למונה זה כיון שמערכת ההפעלה שולחת מספר אירועי MouseMove בעת האתחול.

הוספת אנימציה

הפעל את תוכנית שומר המסך בסביבת הפיתוח של Visual Basic ותראה מסך ריק עד אשר תקיש על מקש או תזיז את העכבר. שומר מסך זה אינו מרגש במיוחד. הוא זקוק לאנימציה שניתן להוסיפה באמצעות פקדי Timer ו-Image. על פקד Timer למדת בפרק 4. המשך והוסף אנימציה. לשומר המסך שלי, יצרתי אנימציה של מרדף מכוניות כמתואר בתרשים 23.3.



תרשים 23.3: טופס שומר המסך מוצג בתצוגת העיצוב עם כל מרכיבי האנימציה

ראה "יצירת אנימציה פשוטה" בפרק 4.

הקוד לסוג זה של אנימציה פשוט מאוד. ודא רק הוספת משפט DoEvents כך שהאירוע MouseMove יוכל להתבצע.

```
Dim CarX As Integer, CarY As Integer

Private Sub Form_Load()
If App.PrevInstance Then End
CheckCommandLine
CarX = Me.Width + 1000
CarY = 0
End Sub

Private Sub tmrMain_Timer ()
        Carx = CarX + 35
        Cary = CarY + 13

        If CarX < LeftBorder - 1000 Then Form_Load

        ImgCar.Top = CarY
        ImgCar.Left = CarX
        ImgBronco.Top = CarY + 230
        ImgBronco.Left = CarX - 7000

        DoEvents
EndSub
```

כמו-כן שים לב כי התוכנית מסתיימת באירוע Load אם מרכיב אחר מופיע. קוד זה חשוב מאוד היות ו-Windows מעלה לפעמים מספר מופעים של שומר המסך.

התקשרות עם Windows

כעת משהושלם שומר המסך נותר רק להוסיף מספר שורות כך שהתוכנית תעבוד כמו שצריך לפעול שומר מסך. כידוע לך הכרטיסיה שומר מסך בתיבת הדו-שיח Display Properties מאפשרת להגדיר ולראות בתצוגה מקדימה את שומר המסך. בכל מקרה כזה, Windows שולחת שורת פקודות שונה לתוכנית שלך. הוסף את הפונקציה CheckCommandLine לטופס שלך כך שיתנהג כמו שצריך.

```
Sub CheckCommandLine ()
        Dim sCmdLine As String
        sCmdLine = Trim$(Lcase$(Command$))

        'RUNNING AS A SCREEN SAVER
        If sCmdLine = "/s" Or sCmdLine = "" Then
                Exit Sub
        End If
```

```
	'RUNNING IN SETUP (CONFIG) MODE
	If sCmdLine= " /c" Then
		MsgBox "Config Screen not Available"
		End
	End IF

	'RUNNING IN PREVIEW MODE
	If Left$(sCmdLine, 2)  = " /p " Then
		'parameter can be retrived with Val(Mid$(sCmdLine,3))
	End
	End If
End Sub
```

לבסוף, כל שנותר הוא להדר את התוכנית, לשנות את סיומת exe לסיומת scr ולהעתיק את התוכנית לתיקיה של Windows. אזי היא אמורה להופיע בצורה אוטומטית ברשימת שומרי המסך הנגישים מתוך כרטיסיית שומר המסך של תיבת הדו-שיח Display Properties.

תוכנית ייצוא טבלאות משרת SQL למסד נתונים של Access

חברות גדולות נוטות להתבסס על סוגים שונים של מאגרי מידע, כשקנה המידה נע ממאגר נתונים על MainFrame ועד למאגר נתונים של משתמש בודד על מחשב PC. יישומים חדשים הדורשים גישה לבסיסי נתונים מופיעים בתדירות גבוהה ויוצרים מערכת מסובכת של תלות הדדית.

לדוגמה, יועץ יכול להיות מועסק בכתיבת יישום המצריך קריאת נתונים קיימים מבסיס נתונים SQL Server. באמצעות ODBC, ניתן בפשטות לאפשר ליישום היועץ להתחבר למאגר הנתונים, אולם במצבים מסוימים מצב זה עלול לגרום לבעיה.

- הממונה על SQL Server צריך להוסיף מידע נוסף על התחברות היישום החדש.
- ביצועי SQL Server יכולים להיפגע כתוצאה מריבוי היישומים.
- שינוי טבלאות ב- SQL Server עלול להביא לצורך בשינוי קוד היישום, מה שעלול להיות מסובך לאחר עזיבת היועץ.

תשובה פשוטה לבעיות אלו היא יצירת קובץ ייצוא על בסיס אוטומטי, ובו תיעזר התוכנית החדשה. מסד נתונים של Access הוא פורמט אידיאלי לקובץ ייצוא זה היות והתוכנה יכולה להשתמש בו בדיוק כמו ב- SQL Server.

בניית התוכנית לדוגמה

במקרה ולא שמת לב, אני אוהב להתבסס על קבצי INI ולכך יש סיבה. אפשר להפוך את התוכנית לשימושית יותר, דבר שישתקף בדוגמה זו היות וקבצי INI מכילים את המידע הדרוש לתוכנית לביצוע פעולת הייצוא. אפשר להדר את התוכנית פעם אחת, ולהשתמש בקבצי INI שונים לפעולות ייצוא שונות. הדבר נעשה בשורת פקודה ומבוצע בצורה הבאה:

```
MDBMAKE C:\Data\REVENUE.INI
MDBMAKE C:\Data\EmployeeID
```

קבצי INI עצמם מכילים מידע לצורך התחברות התוכנית ל- SQL Server ויצירת טבלאות Access. לדוגמה, נתונים בקובץ מציינים את מיקום מסד הנתונים של Access (מסד הנתונים המיועד) ואת מחרוזת ההתחברות לשרת SQL (מסד הנתונים המקורי).

```
DBFile=\\MYSERVER\PUBLIC\HRDATA.MDB
SQLConnect="ODBC:DATABASE=personnel:uid=fred:pwd=garvin: DNS=MYSQLDB"
```

בהמשך, הקטע [Tables] מציין את שם ומספר הטבלאות שתרצה ליצור.

```
[Tables]
Tables=4
Table1SQl=hrinfo
Table1MDB=hrinfo
Table1INI=hrinfo
```

כל טבלה מכילה קטע המציין את המספר וסוג השדות בטבלה. לדוגמה, קטע hrinfo יכול להיראות כך:

```
[hrinfo]
Fields=2
Fd1Name="EmployeeId"
Fd1Type=DOUBLE
Fd2Name="Name"
Fd2Type=TEXT
Fd2Size=40
```

בזמן ההפעלה, התוכנית משתמשת בלולאת For לסרוק כל קטע בקובץ INI בזמן שהוא זקוק למידע על השדה. לדוגמה, לטבלה hrinfo המיוצרת על ידי התוכנית, יש שני שדות: EmployeeID מסוג Double ו-Name מסוג Text בעל 40 תווים.

הבנת התוכנית לדוגמה

לתוכנית הייצוא של SQL יש מספר פונקציות כלליות ללא ממשק משתמש אמיתי, (מלבד הצגת התקדמות התוכנית על המסך). הפונקציות החשובות הן:

- CreateLocalFile - יוצרת את קובץ מסד הנתונים (MDB).
- CreateTable - יוצרת טבלה ריקה במסד הנתונים.
- TransferTable - מעבירה את הנתונים מ- SQL Server לטבלה החדשה במסד הנתונים.
- Main - מפקחת על תהליך התקדמות התוכנית.

פונקציות אלו מוצגות בתוכנית 23.3. בתוכנית יש הסבר על מה שקורה בזמן הפעלתה.

תוכנית 23.3: SQLXPORT.VBP - העברת נתונים מ- SQL Server לתוכנת Access.

```
'Important variables:
Dim dbLocal As Database            'Destination database (Access)
Dim dbSQL As Database              'Source database (MS SQL Server)
Dim sDBLocalPath As String         'Path to destination database
Dim sINIPath As String             'Path to INI File

Sub Main()
        Dim sConnect As String      'SQL Server Connect String
        Dim nTables As Integer      'Number of tables to transfer
        Dim nCurTable As Integer    'Counter for current table
        Dim sSQLTable As String     'Name of the SQL table
        Dim sMDBTable As String     'Name of the Access table
        Dim sSection As String      'INI file [Section]
        Dim nCommand As Integer     'Commands can be run on the
        Dim sCommand As String      'Access database (i.e. create index)

        'GET INI FILE NAME
        sINIPath = App.Path & "\SQLXPORT.ini"
        If Trim$(Command$) <> "" Then sINIPath = Command$

        'READ INFO FROM THE INI FILE
        sDBLocalPath = sGetINIString(sINIPath, "General", "DBFile", "test.mdb")
        sConnect = sGetINIString(sINIPath, "General", "SQLConnect", "ODBC;")
        nTables = CInt(sGetINIString(sINIPath, "Tables", "Tables", "0"))

        'SHOW FORM AND CONNECT TO SQL SERVER
        frmWait.Show
        frmWait.lblWait = "Connecting to SQL Server..."
```

```
        DoEvents
        On Error GoTo MainError
        Set dbSQL = OpenDatabase("", False, True, sConnect)

        'CALL FUNCTION TO CREATE MDB FILE
        frmWait.lblWait = "Creating MDB file..."
        DoEvents
        CreateLocalFile
        DoEvents

        'CREATE TABLES IN THE NEW MDB FILE
        For nCurTable = 1 To nTables
                sSQLTable = sGetINIString(sINIPath, "tables", "table" & nCurTable & _
                        "SQL", "none")
                sMDBTable = sGetINIString(sINIPath, "tables", "table" & nCurTable & _
                        "MDB", "none")
                sSection = sGetINIString(sINIPath, "tables", "table" & nCurTable & "INI", _
                        "none")
                frmWait.lblWait = "Transferring " & sMDBTable
                TransferTable sSQLTable, sMDBTable, sSection
        Next nCurTable

        'THE DATABASE HAS BEEN CREATED, RUN COMMANDS ON IT IF NECESSARY
        nCommand = 0
        sCommand = ""
        Do While sCommand <> "?"
                If nCommand <> 0 Then dbLocal.Execute sCommand
                nCommand = nCommand + 1
                sCommand = sGetINIString(sINIPath, "General", "Command" & _
                        nCommand, "?")
        Loop

        'CLOSE EVERYTHING DOWN AND END
        frmWait.lblWait = "Disconnecting..."
        dbLocal.Close
        dbSQL.Close
        Unload frmWait
        DoEvents
        End

MainError:
        frmWait.Hide
        Screen.MousePointer = vbDefault
        WriteErrMsg "Main - Error " & Err & ": " & Error
        End
End Sub
```

```
Sub TransferTable(sSQLTable As String, sLocalTable As String, sSection As String)

'This function transfers data from SQL server to an Access table

        Dim rstemp As Recordset         'SQL recordset
        Dim aTable As Recordset         'Access table
        Dim nTemp As Integer            '   Counter
        Dim nFields As Integer          '   variables
        Dim nCount As Integer           '
        Dim sSQL As String              'SQL statement
        Dim sTemp As String

On Error GoTo TRTError:

        frmWait.ProgressBar1.Visible = True
        nCount = 0
        frmWait.ProgressBar1.Min = 0

        'The user can either transfer a SQL table as-is,
        'or the results of a query involving multiple tables.
        'This next IF statement determines which and sets up the
        'SQL statement- either "Select * from table" or the
        'user-defined SQL statement.

        sSQL = sGetINIString(sINIPath, sSection, "SQL", "")
        If sSQL = "" Then
                Set rstemp = dbSQL.OpenRecordset("Select Count(*) from " & sSQLTable, _
                        dbOpenSnapshot, dbForwardOnly)
                frmWait.ProgressBar1.Max = CInt(Trim$(" 0" & rstemp.Fields(0)))
                rstemp.Close
                DoEvents
        Else
                nTemp = 2
                sTemp = sGetINIString(sINIPath, sSection, "SQL" & nTemp, "")
                Do While sTemp <> ""
                        If sTemp <> "" Then sSQL = sSQL & sTemp
                        nTemp = nTemp + 1
                        sTemp = sGetINIString(sINIPath, sSection, "SQL" & nTemp, "")
                Loop
                frmWait.ProgressBar1.Max = 2000 'Set arbitrary value on progress bar
        End If
```

```
	'Actually open the recordset
	If sSQL = "" Then
		Set rstemp = dbSQL.OpenRecordset("Select * from " & sSQLTable, _
			dbOpenSnapshot, dbForwardOnly)
	Else
		Set rstemp = dbSQL.OpenRecordset(sSQL, dbOpenSnapshot, _
			dbForwardOnly + dbSQLPassThrough)
	End If

	'Open Local table
	Set aTable = dbLocal.OpenRecordset(sLocalTable, dbOpenTable)
	nFields = rstemp.Fields.Count - 1

	'Transfer each record
	Do While Not rstemp.EOF
		aTable.AddNew
		For nTemp = 0 To nFields
			aTable.Fields(nTemp) = rstemp.Fields(nTemp)
		Next nTemp
		rstemp.MoveNext
		nCount = nCount + 1
		frmWait.ProgressBar1.Value = nCount
		aTable.Update
	Loop
	rstemp.Close
	aTable.Close
	Exit Sub

TRTError:
	WriteErrMsg "TRT-Error " & Err & ": " & Error
	End
End Sub

Sub CreateTable(ByRef tbl As TableDef, sTableName As String, sINISection As String)
'This function creates an empty table in an Access database

	Dim nFields As Integer  'Number of Fields
	Dim Fd() As New Field  'Array of fields
	Dim nCurField As Integer 'Counter for current field
	Dim sTemp As String
```

```
On Error GoTo CRTError:

        nFields = CInt(sGetINIString(sINIPath, sINISection, "Fields", "0"))
        If nFields = 0 Then Exit Sub
        ReDim Fd(1 To nFields)
        tbl.Name = sTableName

        For nCurField = 1 To nFields
                Fd(nCurField).Name = sGetINIString(sINIPath, sINISection, "Fd" & _
                                nCurField & "Name", "ERROR" & nCurField)
                sTemp = sGetINIString(sINIPath, sINISection, "Fd" & nCurField & "Type", _
                        "TEXT")
                Select Case sTemp
                        Case "DOUBLE"
                                Fd(nCurField).Type = dbDouble
                        Case "MEMO"
                                Fd(nCurField).Type = dbMemo
                                'VB4/5 only
                                Fd(nCurField).AllowZeroLength = True
                        Case "BYTE"
                                Fd(nCurField).Type = dbByte
                        Case "INTEGER"
                                Fd(nCurField).Type = dbInteger
                        Case "DATE"
                                Fd(nCurField).Type = dbDate
                                Fd(nCurField).Required = False

                        Case Else 'Text
                                Fd(nCurField).Type = dbText
                                Fd(nCurField).Size = CInt(sGetINIString(sINIPath, _
                                        sINISection, "Fd" & nCurField & "Size", "50"))
                                Fd(nCurField).AllowZeroLength = True
                        End Select

                        tbl.Fields.Append Fd(nCurField)
                Next nCurField
                Exit Sub
CRTError:
        WriteErrMsg "CRT-Error " & Err & ": " & Error
        End
        Exit Sub
End Sub
```

```
Sub CreateLocalFile()
'This procedure creates the MDB file itself
	Dim MainTable() As New TableDef
	Dim sTemp As String
	Dim nTables As Integer
	Dim nCurTable As Integer
	Dim sSQLTable As String
	Dim sMDBTable As String
	Dim sSection As String

On Error GoTo CRLError

	If bFileExists(sDBLocalPath) Then Kill sDBLocalPath
	Set dbLocal = CreateDatabase(sDBLocalPath, dbLangGeneral)

	nTables = CInt(sGetINIString(sINIPath, "Tables", "Tables", "0"))
	ReDim MainTable(1 To nTables)

	For nCurTable = 1 To nTables
		sSQLTable = sGetINIString(sINIPath, "tables", "table" & nCurTable & _
			"SQL", "none")
		sMDBTable = sGetINIString(sINIPath, "tables", "table" & nCurTable & _
			"MDB", "none")
		sSection = sGetINIString(sINIPath, "tables", "table" & nCurTable & "INI", _
			"none")
		CreateTable MainTable(nCurTable), sMDBTable, sSection
		dbLocal.TableDefs.Append MainTable(nCurTable)
	Next nCurTable
	Exit Sub

CRLError:
	WriteErrMsg "CRL-Error " & Err & ": " & Error
	End
End Sub

Private Sub WriteErrMsg(sMessage As String)
	'Should an error occur, this function writes it to
	'another INI file. I did it this way because the
	'program runs as a scheduled process on a remote
	'machine, so no one would be there to answer an error
	'dialog. There is a second VB program that continuously
	'checks the Error.ini and pages me with the error message.

	Dim sErrorINI As String
	sErrorINI = sGetINIString(sINIPath, "General", "ErrorINI", "error.ini")
	writeINIString sErrorINI, "Error", "Message", sMessage
	writeINIString sErrorINI, "Error", "Error", "True"
End Sub
```

```
Sub writeINIString(INI$, App$, key$, newstr$)
        Dim I%
        Dim sTmp As String
         sTmp = ""
        For i% = 1 To Len(newstr$)
                If Mid$(newstr$, i%, 1) <> Chr$(13) And Mid$(newstr$, i%, 1) <> _
                              Chr$(10) Then
                        sTmp = sTmp & Mid$(newstr$, i%, 1)
                Else
                          sTmp = sTmp & " "
                End If
        Next i%
        i% = WritePrivateProfileString(App$, key$, sTmp$, INI$)
End Sub
Function sGetINIString(INI$, App$, key$, Dflt$) As String

       Dim sTmp As String * 256
        Dim n As Integer
        Dim i As Integer

        n = 255
        sTmp = Space$(n + 1)
        i = GetPrivateProfileString(App$, key$, Dflt$, sTmp, n, INI$)
        sGetINIString = Left$(sTmp, i)

End Function
Function bFileExists(sPath As String) As Boolean
        Dim x As Integer
        x = FreeFile

        On Error Resume Next
        Open sPath For Input As #x

        If Err = 0 Then
              bFileExists = True
        Else
                bFileExists = False
        End If

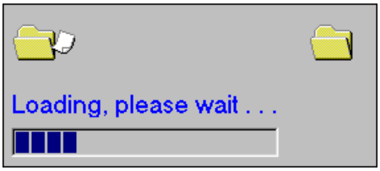
        Close #x

End Function
```

הפונקציה CreateLocalFile מקבלת שם קובץ מהפקודה = Dbpath INI ומשתמשת בפונקציה CreateDatabase של Visual Basic ליצירת קובץ חדש (וריק) מסוג MDB.

בהמשך התוכנית משתמשת בלולאות For לקריאת תוכן הטבלאות קטע [TABLES] וקריאה בצורה מחזורית לשיגרה CreateTable. שיגרה זו, בתורה, סורקת בצורה מחזורית את הקטע tablenames בקובץ INI, ויוצרת שדות בטבלה החדשה.

לאחר יצירת טבלה, השיגרה TransferTable מטפלת בהעברת הנתונים מטבלת SQL Server לאחת הטבלאות החדשות. השיגרה יוצרת תחילה רשימת נתונים מטבלת SQL Server ואז סורקת כל שדה ברשימת הנתונים, ומעבירה את הנתונים לטבלה החדשה. בזמן התהליך, המשתמש רואה מד התקדמות, כמתואר בתרשים 23.4, כדי להראות שמשהו מתרחש.



תרשים 23.4: תוכנית ייצוא הנתונים מציגה מד התקדמות בעת פעולתה

API של Windows ליצירת תמונות שקופות

אחד הרכיבים של Win32 API הוא GDI (ממשק משתמש גרפי). כיון שפונקציות API אלו מורכבות מאוד, בלתי יציבות, ונוטות ליצור שגיאות GPF, Microsoft הוציאה את רובן מתוך Visual Basic. למרות שנטישת API מגינה מפני מורכבות ומייצבת את התוכנית, היא מגבילה מאוד את היכולת לעשות דברים "מגניבים" שמשתמשים רבים מצפים להם. Visual Basic 4 פתרה את הבעיה במידה מסוימת על ידי הכנסת יכולות חדשות כמו הפונקציה PaintPicture והפקד ImageList אבל אלה עדיין לא הספיקו. פירוש הדבר שמתי שהוא בעתיד הקרוב תמצאו עצמכם קוראים לפונקציות GDI API מתוך היישום שלכם. כדי להדגים את השימוש בפונקציות GDI API נכתוב פונקציה "מגניבה" בשם TransparentPaint.



הערה:

גרסת השיגרה TransparentPaint אותה רואים בפרק זה, הינה גרסת Win32 של הקוד TransparentPaintBlt (שנכתב על ידי מייק בונד) שהופיע במקור בתוך Microsoft KB KnowledgeBase - מאמר מספר Q94961. גירסה זו עברה שינויים רבים ומכילה מספר רב של הערות חדשות.

TransparentPaint המוצגת בתוכנית 23.4 מתוכננת לטפל בתמונת מפת סיביות כבסמל, כאשר מציבים אותה של המשטח. אפשר לעצב חלק של הסמל להיות שקוף אבל לא ניתן לעשות זאת עם תמונת מפת סיביות. כדי להשלים פעולה מסובכת זו עליך ליצור סדרת תמונות מפת סיביות זמניות ולעשות שינויים רק בזיכרון. למרות שתפיסה אבסטרקטית זו יכולה להיות מסובכת, ההערות מסייעות להבין מה קורה בכל שלב.

```
'*******************************************************************
'Paints a bitmap on a given surface using the surface backcolor
'everywhere lngMaskColor appears on the picSource bitmap
'*******************************************************************
Sub TransparentPaint(objDest As Object, picSource As StdPicture,  _
        lngX As Long, lngY As Long, ByVal lngMaskColor As Long)
        '**************************************************************
        ' This sub uses a bunch of variables, so lets declare and explain
        ' them in advance...
        '**************************************************************
        Dim lngSrcDC As Long            'Source bitmap
        Dim lngSaveDC As Long  'Copy of Source bitmap
        Dim lngMaskDC As Long  'Monochrome Mask bitmap
        Dim lngInvDC As Long            'Monochrome Inverse of Mask bitmap
        Dim lngNewPicDC As Long         'Combination of Source & Background bmps

        Dim bmpSource As BITMAP         'Description of the Source bitmap

        Dim hResultBmp As Long          'Combination of source & background
        Dim hSaveBmp As Long  'Copy of Source bitmap
        Dim hMaskBmp As Long            'Monochrome Mask bitmap
        Dim hInvBmp As Long             'Monochrome Inverse of Mask bitmap

        Dim hSrcPrevBmp As Long         'Holds prev bitmap in source DC
        Dim hSavePrevBmp As Long        'Holds prev bitmap in saved DC
        Dim hDestPrevBmp As Long        'Holds prev bitmap in destination DC
        Dim hMaskPrevBmp As Long        'Holds prev bitmap in the mask DC
        Dim hInvPrevBmp As Long         'Holds prev bitmap in inverted mask DC

        Dim lngOrigScaleMode&  'Holds the original ScaleMode
        Dim lngOrigColor&               'Holds original backcolor from source DC
        '**************************************************************
        ' Set ScaleMode to pixels for Windows GDI
        '**************************************************************
        lngOrigScaleMode = objDest.ScaleMode
        objDest.ScaleMode = vbPixels
        '**************************************************************
        ' Load the source bitmap to get its width (bmpSource.bmWidth)
        ' and height (bmpSource.bmHeight)
        '**************************************************************
        GetObject picSource, Len(bmpSource), bmpSource
        '**************************************************************
        ' Create compatible device contexts (DC's) to hold the temporary
        ' bitmaps used by this sub
        '**************************************************************
```

```
lngSrcDC = CreateCompatibleDC(objDest.hdc)
lngSaveDC = CreateCompatibleDC(objDest.hdc)
lngMaskDC = CreateCompatibleDC(objDest.hdc)
lngInvDC = CreateCompatibleDC(objDest.hdc)
lngNewPicDC = CreateCompatibleDC(objDest.hdc)
'**************************************************************
' Create monochrome bitmaps for the mask-related bitmaps
'**************************************************************
hMaskBmp = CreateBitmap(bmpSource.bmWidth, bmpSource.bmHeight, 1, 1, _
        ByVal 0&)
hInvBmp = CreateBitmap(bmpSource.bmWidth, bmpSource.bmHeight, 1, 1, _
        ByVal 0&)
'**************************************************************
' Create color bitmaps for the final result and the backup copy
' of the source bitmap
'**************************************************************
hResultBmp = CreateCompatibleBitmap(objDest.hdc, bmpSource.bmWidth, _
bmpSource.bmHeight)
hSaveBmp = CreateCompatibleBitmap(objDest.hdc, bmpSource.bmWidth, _
bmpSource.bmHeight)
'**************************************************************
' Select bitmap into the device context (DC)
'**************************************************************
hSrcPrevBmp = SelectObject(lngSrcDC, picSource)
hSavePrevBmp = SelectObject(lngSaveDC, hSaveBmp)
hMaskPrevBmp = SelectObject(lngMaskDC, hMaskBmp)
hInvPrevBmp = SelectObject(lngInvDC, hInvBmp)
hDestPrevBmp = SelectObject(lngNewPicDC, hResultBmp)
**************************************************************
' Make a backup of source bitmap to restore later
'**************************************************************
BitBlt lngSaveDC, 0, 0, bmpSource.bmWidth, bmpSource.bmHeight, lngSrcDC,_
        0, 0, vbSrcCopy
'**************************************************************
' Create the mask by setting the background color of source to
' transparent color, then BitBlt'ing that bitmap into the mask
' device context
'**************************************************************
lngOrigColor = SetBkColor(lngSrcDC, lngMaskColor)
BitBlt lngMaskDC, 0, 0, bmpSource.bmWidth, bmpSource.bmHeight, lngSrcDC, _
        0, 0, vbSrcCopy
'**************************************************************
' Restore the original backcolor in the device context
'**************************************************************
SetBkColor lngSrcDC, lngOrigColor
```

```
'**************************************************************
' Create an inverse of the mask to AND with the source and combine
' it with the background
'**************************************************************
BitBlt lngInvDC, 0, 0, bmpSource.bmWidth, bmpSource.bmHeight, lngMaskDC, _
        0, 0, vbNotSrcCopy
'**************************************************************
' Copy the background bitmap to the new picture device context
'  to begin creating the final transparent bitmap
'**************************************************************
BitBlt lngNewPicDC, 0, 0, bmpSource.bmWidth, bmpSource.bmHeight,  _
        objDest.hdc, lngX, lngY, vbSrcCopy
'**************************************************************
' AND the mask bitmap with the result device context to create
' a cookie cutter effect in the background by painting the black
' area for the non-transparent portion of the source bitmap
'**************************************************************
BitBlt lngNewPicDC, 0, 0, bmpSource.bmWidth, bmpSource.bmHeight, _
        lngMaskDC, 0, 0, vbSrcAnd
'**************************************************************
' AND the inverse mask with the source bitmap to turn off the bits
' associated with transparent area of source bitmap by making it black
'**************************************************************
BitBlt lngSrcDC, 0, 0, bmpSource.bmWidth, bmpSource.bmHeight, _
        lngInvDC, 0, 0, vbSrcAnd
'**************************************************************
' XOR the result with the source bitmap to replace the mask color
' with the background color
'**************************************************************
BitBlt lngNewPicDC, 0, 0, bmpSource.bmWidth, bmpSource.bmHeight, _
        lngSrcDC, 0, 0, vbSrcPaint
'**************************************************************
' Paint the transparent bitmap on source surface
'**************************************************************
BitBlt objDest.hdc, lngX, lngY, bmpSource.bmWidth, _
        bmpSource.bmHeight, lngNewPicDC, 0, 0, vbSrcCopy
'**************************************************************
' Restore backup of bitmap
'**************************************************************
BitBlt lngSrcDC, 0, 0, bmpSource.bmWidth, bmpSource.bmHeight, _
        lngSaveDC, 0, 0, vbSrcCopy
'**************************************************************
' Restore the original objects by selecting their original values
'**************************************************************
SelectObject lngSrcDC, hSrcPrevBmp
SelectObject lngSaveDC, hSavePrevBmp
SelectObject lngNewPicDC, hDestPrevBmp
SelectObject lngMaskDC, hMaskPrevBmp
```

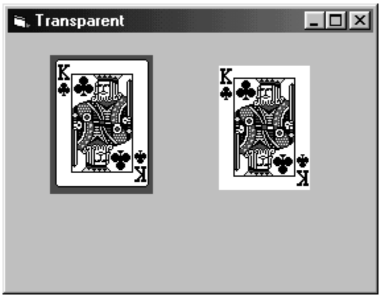
```
        SelectObject lngInvDC, hInvPrevBmp
        '*************************************************************
        ' Free system resources created by this sub
        '*************************************************************
        DeleteObject hSaveBmp
        DeleteObject hMaskBmp
        DeleteObject hInvBmp
        DeleteObject hResultBmp
        DeleteDC lngSrcDC
        DeleteDC lngSaveDC
        DeleteDC lngInvDC
        DeleteDC lngMaskDC
        DeleteDC lngNewPicDC
        '*************************************************************
        ' Restores the ScaleMode to its original value
        '*************************************************************
        objDest.ScaleMode = lngOrigScaleMode
End Sub
```

מסיבות פשטות החסרתי את הצהרות API מתוכנית 23.4 ויכולתי לפרוס על גבי עמודים רבים את ההסברים על המתרחש בכל צעד של TransparentPaint. לא אעשה זאת היות וקטע זה מכיל את אותן הערות שעשיתי ברשימתי. כמו-כן לעקוב אחר תוכנית זו, עלול להיות קשה יותר אם היא מחולקת למקטעים רבים יותר. לאחר שתקרא את ההערות בקטע זה, אני ממליץ לעקוב אחר פרויקט TransparentPaint שאפשר להורידו מאתר של Macmillin's בכתובת www.mcp.com/info. הקריאה תסייע לך להבין מה קורה בכל שלב.

למרות שקשה לעקוב אחרי שיגרה כמו TransparentPaint השימוש בה קל מאוד. תוכנית 23.5 טוענת קובץ מפת סיביות מקובץ מקור וצובעת אותו בפינה השמאלית העליונה על ידי שימוש TransparentPaint. בהמשך, היא צובעת את התמונה על ידי PaintPicture. המשתנה האחרון Visual BasicGreen מורה ל-TransparentPaint להחליף כל סיבית בתמונה שהיא ירוקה בצבע הרקע של הטופס. התוצאה מוצגת בתרשים 23.5.

תוכנית 23.5: TRANSPAR.TXT - שימוש בשיגרה TransparentPaint.

```
' *********************************************************************
' Transparent.frm - Demonstrates how to use basTransparent's
'       TransparentPaint using a bitmap from a resource file.
' *********************************************************************
Option Explicit
' *********************************************************************
' Gets a stdPicture handle by loading a bitmap from a resource file and paints it
' transparently on the form by using Gray as a mask color.
' *********************************************************************
Private Sub cmdPaintTransBmp_Click()
       TransparentPaint Me, LoadResPicture(103, 0), 0, 0, OBColor(7)
End Sub
```



תרשים 23.5: TransparentPaint חייבת להופיע בישומי המולטימדיה שלכם

נסה להחליף את קובץ המקור של פרויקט זה בקובץ משלך והרגש כיצד עובדת TransparentPaint. כמו-כן נסה להשתמש במסכות צבע ובתמונות שונות. מהיום לעולם לא תצטרך לכתוב יישום שנראה נחות יותר משום שאינו משתמש בתמונות שקופות.

מכאן...

בפרק זה ראית מיגוון פרויקטים. אם תזדקק למידע נוסף על הנושאים שנדונו, פנה לפרקים הבאים:

- ללמוד על קריאות API ראה פרק 20.
- למידע על קבצי INI פנה לפרק 21.

חלק 6

Visual Basic ומסדי נתונים



24

יסודות מסד הנתונים

מה בפרק?

- עיצוב מסדי נתונים
- מימוש העיצוב שלך
- השימוש ב- Visual Data Manager
- יצירת מסד נתונים בעזרת כלים אחרים
- מדוע כדאי להשתמש בתוכנה במקום ב-Visual Data Manager

ניתן לומר בביטחון שכל תוכנת מחשב יישומית עובדת עם **נתונים** (Data) מסוג כלשהו. הנתונים האלה מאוחסנים לרוב **במסד נתונים** (Database) או בכמה מסדי נתונים. Visual Basic מאפשרת לך ליצור תוכנות חזקות לניהול מידע, תוך השקעה של מעט תכנון ומאמץ. המרכיב החשוב ביותר בעיצוב מסדי נתונים הוא הגדרת מבנה הנתונים. מסד נתונים שלא עוצב נכון עלול להעמיד מכשולים גם בפני פיתוח תוכניות מתקדמות מאוד. מאידך, שימוש במסד נתונים שעוצב כראוי יכול להקל מאוד על חיי המפתח.

יצירת **מבנה נתונים** (Data Structure) מאורגן היטב דורשת ממך ללמוד את אופן ביצוע שתי משימות. ראשית, עליך ללמוד כיצד לעצב מסדי נתונים. בשלב העיצוב, עליך להחליט אילו סוגי נתונים ייכללו במסד הנתונים, ואיך הם יאורגנו במסגרתו. אחר כך, תידרש ללמוד כיצד לתרגם את העיצוב שלך למסד נתונים ממשי. תוכל לבצע פעולה זאת במיגוון דרכים. בפרק זה נבחן שיקולים שונים הקשורים לעיצוב מסדי נתונים, ומבנים של מסדי נתונים. את המושגים שתלמד בפרק זה תוכל ליישם על מסדי נתונים מכל הסוגים, ולא רק על מסדי נתונים שתבנה במסגרת יישומים ב- Visual Basic.

עיצוב מסד נתונים

בדומה לפעולות אחרות, גם בניית מסד נתונים מתחילה בשלב העיצוב. כפי שלא היית מתחיל לבנות בית מבלי לתכנן אותו קודם, או שלא היית מבשל תבשיל לאניני טעם מבלי לעיין במרשם, כך לא כדאי שתתחיל בבניית מסד נתונים מבלי לעצב אותו. כפי שקורה בביצוע פרויקטים מסוגים אחרים, גם בעת בניית מסדי נתונים השקעה בתכנון ועיצוב היא שלב ראשון בביצוע פרויקט מוצלח.

בעת עיצוב יישום לניהול מסד נתונים (Database Application) תידרש, בנוסף לתכנות שגרות התוכנית באופן שיאפשר לך להשיג ביצועים אופטימליים, גם להקדיש תשומת לב למבנה הלוגי והפיסי בו יאוחסן המידע, אליו מתייחסת התוכנית. תהליך עיצוב נכון של מסד נתונים מאפשר לך להשיג את היעדים הבאים:

- לקצר ככל האפשר את זמני החיפוש הדרושים לשם איתור רשומות מסוימות.
- לאחסן מידע באופן יעיל ביותר, כדי למנוע ממסד הנתונים לגדול למימדים מוגזמים.
- להקל על ביצוע פעולות לעדכון מידע.
- להקנות למסד הנתונים מבנה גמיש, כדי לאפשר הוספת פונקציות חדשות העשויות להידרש לתוכנית.

יעדי עיצוב

בעת הגדרת עיצוב מסד הנתונים, תידרש להתייחס למספר יעדים שונים. עליך לשאוף להשיג כמה שיותר יעדים, אך במקרים מסוימים השגת יעד מסוים תמנע השגת יעד אחר כלשהו. יעדי העיצוב העיקריים הם:

- מניעת אחסון נתונים מיותרים.
- אפשרות לאיתור מהיר של כל **רשומה** (Record).
- הגדרת מבנה שיקל על הכנסת שיפורים במסד הנתונים.
- שמירה על ניהול נוח של מסד הנתונים בכל שלב.

פעולות חשובות בעיצוב מסד נתונים

יצירת מסד נתונים טוב כרוכה בביצוע שבע הפעולות המפורטות להלן:

- הגדרת מבנה היישום.
- הגדרת הנתונים הדרושים לצורך פעולת היישום.
- ארגון הנתונים ב**טבלאות** (Tables).
- הגדרת מערכת ה**קשרים** (Relationships) בין הטבלאות.
- הגדרת **אינדקסים** (Indexes) **וכללי אימות** (Validation Rules) עבור הנתונים.
- יצירה ואחסון **שאילתות** (Queries) הדרושות ליישום.
- בחינת תהליך העיצוב.

כעת, נסקור בקצרה את שתי הפעולות הראשונות שצוינו ברשימה. ראשית נסקור את פעולת העיצוב היישום. בעת הגדרת מבנה היישום, תידרש תחילה להגדיר את הפעולות שעל היישום לבצע. לדוגמה, אם אתה מנהל את רשימת של מועדון חברים, יהיה עליך לנהל רשימת מספרי הטלפון ורשימת כתובות למשלוח דואר. כאשר אתה מגדיר את הפעולות שתבוצענה על ידי היישום, עליך להגדיר **מפרט פונקציונלי** (Functional Specification) שלו. כאשר אתה מפתח יישום, סביר להניח כי ברורות לך המשימות אשר אותו יישום אמור לבצע, אך למרות זאת כדאי שתכתוב מפרט מסודר של היישום. המפרט הכתוב יסייע לך בהתמקדות בפעולות שאתה מעוניין כי התוכנית תבצע, יסייע להגיע להסכמה מול הגורמים עבורם מיועדת התוכנית ויועיל גם בהגדרת לוחות זמנים לביצוע הפרויקט.

שיחות עם הגורמים שעבורם מיועד היישום הן הדרך הטובה לקבלת רקע על המשימות אותן הוא מיועד לבצע. ראשית, יהיה עליך לברר האם כבר עומדת לרשותם מערכת ישנה, אותה הם מעוניינים להחליף והאם יש דוחות מסוימים שהם מעוניינים להפיק. בשלב הבא כדאי לשאול שאלות רבות ככל האפשר אשר תסייענה לך להבין מהן המשימות אותן מייעד הלקוח ליישום.

לאחר הגדרת המפרט הפונקציונלי, אפשר להתחיל בהגדרת הנתונים הדרושים. בעת יצירת יישום לניהול מועדון חברים, הידיעה שעליך להפיק רשימות כתובות ומספרי טלפון מכוונת אותך לכלול במסד הנתונים כתובת כל חבר ומספר הטלפון שלו. הידיעה כי מיון דואר על פי מספרי מיקוד תסייע להאצת העברתו אל החברים ואף תאפשר ליהנות מתעריפים מיוחדים של דיוור מהיר, תעודד אותך להגדיר אינדקס או שאילתה אשר ימיינו את רשימת הכתובות על-פי מיקוד. דוגמה זו מאפשרת לך לראות שעיצוב נכון של היישום לא רק מגדיר עבורך את המידע שיידרש לך, אלא גם מרכיבים אחרים של היישום.

ארגון המידע

הגדרת אופן אחסון המידע במסד הנתונים היא אחת הפעולות החשובות ביותר בעיצוב מסד נתונים באיכות גבוהה. עיצוב נכון של מסד נתונים כרוך באחסון הנתונים באופן המקל על **אחזור** (Retrieve) הנתונים ועל תחזוקת מסד הנתונים. הנתונים המאוחסנים במסד נתונים מאוחסנים ב**טבלה** (Table) או במספר טבלאות. ביישומים רבים של ניהול מסדי נתונים ניתן להגיע לניהול מידע יעיל, על ידי אחסון הנתונים במספר טבלאות שונות והגדרת **קשרים** (Relationships) בין אותן טבלאות. בסעיפים הבאים תלמד כיצד לקבוע אילו סוגי נתונים ישויכו לכל טבלה במסד הנתונים.

טבלאות כנושאים

טבלה (Table) היא אוסף נתונים אשר מתייחסים לנושא מסוים. תוכל לקבוע האם כדאי לאחסן פריט מידע מסוים בטבלה מסוימת על ידי כך שתתייחס לנושא העיקרי של הטבלה. לדוגמה, אם מועדון מסוים מעוניין לנהל מעקב אחר מידע הקשור בחברי המועדון ובעובדי המועדון, הנהלת המועדון עשויה להתפתות לאחסן את המידע הנוגע לחברי המועדון והמידע הנוגע לעובדי המועדון באותה טבלה (זאת מפני שחברים ועובדים הם אנשים). אך כדאי שנתייחס לרגע לסוגי הנתונים שתידרש לאחסן עבור כל אחת מהקבוצות. לשתי הקבוצות אומנם נדרשים נתונים כגון שמות, כתובות ומספרי תעודת זהות. אך לקבוצת העובדים יידרשו גם נתונים על ניכויי מס ועל הגדרת התפקיד. אם תכלול החברים והעובדים בטבלה אחת, תיצור מצב שבו יהיו שדות ריקים ברשומות החברים. תידרש גם להוסיף לטבלה שדה שיאפשר להבדיל בין עובדים לבין חברים. כבר ברור שאחסון משותף של נתוני העובדים והחברים בטבלה אחת יגרום לבזבוז רב. האחסון המשותף גם יגרום להאטה בביצוע פעולות המתייחסות לעיבוד הרשומות הנוגעות למגזר אחד בלבד, זאת מכיון שהתוכנית תידרש לדלג על קבוצה שלמה של רשומות הכלולות במסד הנתונים. בתרשים 24.1 מוצגת טבלה בה מאוחסנים נתוני עובדים וחברים. בתרשים 24.2 ניתן לראות את הצמצום במספר השדות הכלולים בטבלה, המתאפשר על ידי אחסון נתוני החברים בטבלה נפרדת.

תשומת לב לנושא אליו מתייחסת הטבלה מאפשרת לך לקבוע בקלות רבה יותר האם פריט מידע מסוים צריך להיכלל בטבלה זו או אחרת. אם הכללתו של אותו פריט תגרום לניפוח מיותר של הטבלה, סביר להניח כי מקומו בטבלה אחרת.

טבלה : Names

שם משפחה	שם פרטי	כתובת	עיר	מדינה	מיקוד	עובד חברה	תעודת-זהות	תפקיד	משכורת
כהן	יעקב	הזוית 6	ת"א	ישראל	65210	לא			
גולדשטיין	משה	הרב-קוק 13	באר-שבע	ישראל	78541	כן	054487542	גנן	20 ש"ח
קינן	מאיר	ההסתדרות 32	חיפה	ישראל	98547	כן	981154732	מנכ"ל	ע"פי חוזה
וקנין	יקר	האבות 20	ראשל"צ	ישראל	75305	לא			
פלו	יוני	ויו נגוף 124	ונ"א	ישו אל'	68100	כן	057614565	אוו אי ו נש	35 ש"ח

רשומה: 6 מתוך 6

תרשים 24.1: אחסון נתוני העובדים ונתוני החברים ביחד, בטבלה אחת, גורם לבזבוז רב

טבלה : Members

שם משפחה	שם פרטי	כתובת	עיר	מדינה	מיקוד
כהן	יעקב	הזוית 6	ת"א	ישראל	65210
גולדשטיין	משה	הרב-קוק 13	באר-שבע	ישראל	78541
קינן	מאיר	ההסתדרות 32	חיפה	ישראל	98547

רשומה: 4 מתוך 4

תרשים 24.2: אחסון נתוני החברים בטבלה נפרדת מאפשר לכלול בה רק את השדות הרלוונטיים לחברים והוא יעיל הרבה יותר

נרמול נתונים

נרמול נתונים (Data Normalization) הוא תהליך גריעת מידע מיותר ממסד נתונים. באופן תיאורטי, נרמול נתונים אופטימלי יוצר מצב בו כל פריט מידע מופיע פעם אחת בלבד במסד הנתונים בו הוא מאוחסן. באופן מעשי, קשה להגיע לתוצאות אלו אך יש לשאוף אליהן.

לשם הדגמת הנושא, נתייחס למערכת טיפול בהזמנות לקוחות. הנתונים המעניינים אותך בכל הזמנה הם: מספר פריט (Itemno), תיאור הפריט (Description), מספר הזמנה (Orderno), תאריך הזמנה (Order Date), שם הלקוח המזמין (Last Name ו-First Name), כתובתו (Address) ומספר הטלפון שלו (Phone). אם תאחסן את כל המידע הזה בטבלה אחת, הטבלה תיראה כמו זו המוצגת בתרשים 24.3:

טבלה : CustItem

פריט	תאור	מס הזמנה	תאריך הזמנה	מס לקוח	שם משפחה	שם פרטי	טלפון
1001	דג חרב	101	9/4/94	3	קינן	מאיר	7531901
1003	פרות ים	101	9/4/94	3	קינן	מאיר	7531901
1005	דג חרב	102	9/5/94	1	פלד	יוסי	5557877
1010	צדפות	102	9/5/94	1	פלד	יוסי	5557877
1001	צדפות	103	9/5/94	2	וקנין	יקיר	9660495
1005	פרות ים	104	9/9/94	3	קינן	מאיר	7556220

רשומה: 7 מתוך 7

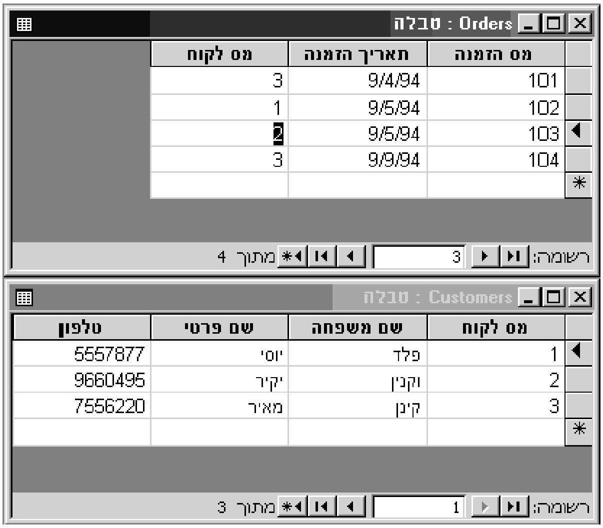
תרשים 24.3: נתונים שלא עברו תהליך נורמליזציה גורמים לטבלאות גדולות ולא יעילות

ניתן לראות כי נתונים רבים הכלולים בטבלה חוזרים על עצמם. העובדה שנתונים חוזרים על עצמם יוצרת שתי בעיות. הבעיה הראשונה היא בזבוז שטח אחסון והיא נובעת מהעובדה שאותם נתונים מאוחסנים בזיכרון כמה פעמים. הבעיה השנייה נוגעת לאמינות (Accuracy) ועדכניות (Currency) המידע. לדוגמה, אם מספר הטלפון של לקוח כלשהו השתנה, תידרש לשנות את המספר בכל הרשומות המתייחסות לאותו לקוח, כאשר תמיד קיימת אפשרות שתדלג בטעות על רשומה כלשהי. בתרשים 24.3 ניתן לראות כי מספר הטלפון של מאיר קינן עודכן ברשומה האחרונה, אך לא עודכן בשתי הרשומות הראשונות. כך שאם עובד כלשהו יעיין באחת משתי הרשומות הראשונות, יוצג לפניו מספר טלפון לא עדכני.

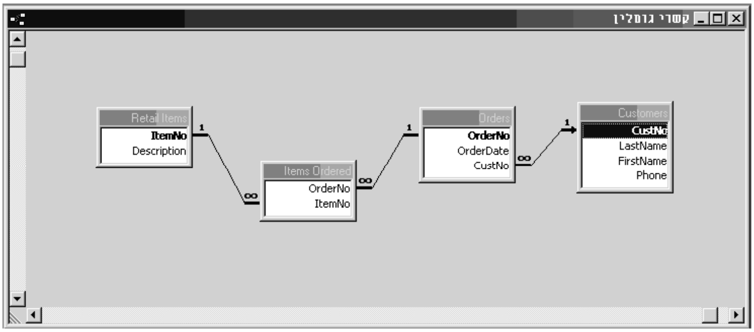
אחסון נתוני הלקוחות בטבלה אחת ונתוני ההזמנות בטבלה אחרת, הינה דרך טובה יותר לטיפול במידע המוצג בדוגמה לעיל. תוכל להגדיר מזהה ייחודי עבור כל לקוח ולכלול את המזהה בטבלת ההזמנות, כדי לזהות את הלקוח. שיטה זו מניבה שתי טבלאות נפרדות, בעלות מבנה המוצג בתרשים 24.4.

מבנה זה, מאפשר לך לרכז את כל נתוני הלקוח במקום מסוים, כך שכדי לשנות נתון כלשהו, לדוגמה מספר טלפון חדש, יהא עליך לגשת למקום אחד בלבד.

תוכל ליישם תהליך דומה גם עבור נתוני הפריטים שנמכרו (Items Sold) ונתוני ההזמנות (Order Information). אומנם מדובר ביצירת ארבע טבלאות שונות, אך תוך ייעול משמעותי של תהליכי העבודה ועיבוד הנתונים. שיטה זו מבטיחה את אמינות המידע ומאפשרת לבצע שינוי פעם אחת ובמקום אחד בלבד. מבנה נתונים מסוג זה מוצג בתרשים 24.5. במסגרת המבנה בן ארבע הטבלאות, טבלאות Orders ו-Items Ordered מיועדות לקשר בין לקוחות לפריטים אותם הזמינו. הטבלה Items Ordered כוללת רשומה אחת עבור כל פריט שנכלל בהזמנה נתונה. הטבלה Orders מקשרת את הפריטים עם תאריך ההזמנה ועם הלקוח שביצע את ההזמנה.



תרשים 24.4: נרמול הנתונים בטבלאות הלקוחות וההזמנות מונע מצבים של מידע מיותר



תרשים 24.5: נרמול אופטימלי של הטבלאות מאפשר להגיע לרמת יעילות גבוהה ביותר

העברת מידע מטבלה אחת לאחרת דורשת ממך ביצוע פעולות מעקב אחר **קשרים** (Relationships) בין טבלאות. את הפעולות האלו תוכל לבצע בעזרת **מפתח** (Key) נתונים. לדוגמה, לטבלה Customers שדה בשם CustNo. גם בטבלה Orders שדה בשם CustNo. הטבלאות **מקושרות** (Linked) ביניהן באמצעות שדות אלה. אם התוכנית תידרש לאתר מידע הנוגע ללקוח אשר ביצע הזמנה מסוימת, היא תוכל לאתר את הרשומה הרלוונטית בקלות, תוך שימוש בשדה המשותף CustNo.

טבלאות צאצא וטבלאות בדיקת מידע

ניתן לבצע נרמול נתונים גם על ידי יצירת טבלאות המכונות טבלאות צאצא. **טבלת צאצא** (Child Table) היא טבלה בה כל הפריטים חולקים ביניהם נתון משותף כלשהו, המאוחסן בטבלה אחרת. רשימת בני אותה משפחה היא דוגמה אפשרית ופשוטה לטבלת צאצא. בני המשפחה חולקים שם משפחה משותף, כתובת משותפת ומספר טלפון משותף, אך לכל אחד מבני המשפחה יש שם פרטי שונה. הטבלה המכילה את המידע המשותף קרויה **טבלת אב** (Parent Table) ואילו הטבלה המכילה את השמות הפרטיים של בני המשפחה היא טבלת הצאצא. בתרשים 24.6 מוצגות טבלת אב וטבלת הצאצא הקשורה אליה.

טבלת בדיקת מידע (Lookup Table) היא דרך נוספת לאחסון מידע בצורה יעילה, תוך שיפור אמינותן של פונקציות להזנת מידע. טבלת בדיקת מידע משמשת לרוב לאחסון פריטי מידע העונים על חוק אימות מסוים (לדוגמה, קיצורי שמות ערים או מדינות). כאשר משתמש מזין פריט מידע מסוג שהוגדר עבור טבלת בדיקת המידע, התוכנית מעיינת בטבלה כדי לוודא כי הערך שהוזן על ידי המשתמש אכן כלול בטבלה.

טבלאות בדיקת מידע יכולות לשמש גם במסגרת פעולות לנרמול מידע. אם אתה מנהל רשימת נמענים ארוכה, הרי סביר כי רשומות רבות בה תכלנה ערכים זהים בשדות העיר והמדינה. במקרה זה, תוכל להשתמש בטבלת המיקוד כדי לאחסן עיר ומדינה על-פי מיקוד (זכור שמספר מיקוד מסוים תואם לצמד אחד של עיר ומדינה). כדי שתוכל להשתמש בטבלת המיקוד ביעילות, יהיה עליך לבנות את טבלת הכתובות כך שתכיל רק שדות מיקוד, ללא שמות העיר והמדינה בהן מצויה הכתובת. בעת הזנת הנתונים תוכל להורות לתוכנית לבדוק כל מספר מיקוד שתזין, כדי לוודא שהוא חוקי.



טבלה : Family Info

מס משפחה	שם משפחה	כתובת	עיר	מיקוד	טלפון
1	פלד	דיזינגוף 124	ת"א	68100	5557877
2	וקנין	האבות 20	ראשל"צ	75305	9660495
3	קינן	ההסתדרות 32	חיפה	98547	7556220

רשומה: 4 מתוך 4



טבלה : Family Members

מס משפחה	שם פרטי	יום-הולדת
1	יוסי	1/1/70
1	תומר	5/4/74
1	דפנה	7/3/75
1	מירב	9/5/78
2	יקיר	27/1/77
2	נעמה	4/8/80
2	דני	8/12/83
3	הילה	9/7/79
3	אורית	4/4/74
3	ליאת	6/7/75

רשומה: 11 מתוך 11

תרשים 24.6: השימוש בטבלאות אב ובטבלאות צאצא הוא דרך לנרמול נתונים

כללים לארגון טבלאות

אומנם, לא קיימים כללים מוחלטים המגדירים אילו סוגי נתונים צריכים להיכלל באילו טבלאות, אך להלן מספר קווים מנחים אשר מומלץ כי תיישם אותם בעת עיצוב מסדי נתונים בתוכניותיך:

- הגדר נושא מסוים עבור כל טבלה והקפד כי כל המידע המאוחסן בטבלה אכן מתייחס לנושא המשותף שהגדרת.
- אם מספר רשומות באותה טבלה נותרו ריקות במכוון, פצל את הטבלה לשתי טבלאות דומות (זכור את הדוגמה של נתוני העובדים והחברים).
- אם מידע מסוים חוזר על עצמו במספר רשומות, העבר מידע זה לטבלה אחרת והגדר קשר בין הטבלאות.
- תופעה של שדות אשר חוזרים על עצמם היא אינדיקציה לצורך בטבלת צאצא. לדוגמה, אם הטבלה שלך כוללת שדות כגון Item1, Item2 ו-Item3, העבר את נתוני הפריטים אל טבלת צאצא, שתקושר אל טבלת האב.
- היעזר בטבלאות בדיקת מידע לצורך צמצום נפח המידע והגברת אמינות הפעולות להזנת הנתונים.
- אל תאחסן בטבלאות פריטי מידע שניתן לחשב אותם תוך שימוש בערכים אחרים הכלולים בטבלה.

הערה:

זכור שהכללים שהובאו כאן אינם "תורה מסיני", וייתכנו מצבים בהם תידרשנה סטיות מהם.

שיקולי ביצועים

אחת הסיבות השכיחות לסטייה מהכללים שפורטו כאן היא כדי לאפשר שיפור בביצועים. לדוגמה, אם לצורך חישוב סך המכירות הכולל שבוצע על ידי איש מכירות מסוים תידרש לסכם אלפי רשומות, כדאי שתכלול בטבלה שדה שיכיל את סך המכירות, ויתעדכן בכל פעם שאותו איש מכירות יבצע מכירה. הוספת שדה כזה תמנע צורך לבצע חישובים רבים בכל פעם שיש להציג דוח. זו פעולה אשר תאיץ את יצירת הדוח באופן ניכר. אך עליך לוודא שהעדכון שדה סך המכירות יתבצע באופן עקבי ואמין.

הצורך בפתיחת מספר גדול של טבלאות בו-זמנית היא סיבה נוספת לסטייה מהכללים שפורטו כאן. עקב העובדה שלכל טבלה נדרשים זיכרון ומשאבי מערכת יקרים, מצב בו הרבה טבלאות תהינה פתוחות ברגע נתון, עלול להאט מאוד את פעולת היישום שלך.

סטייה מהכללים אלה תגרום לשתי תופעות נלוות. ראשית, הגדלת מסד הנתונים כתוצאה מצורך באחסון נתונים מיותרים. שנית, ייתכן מצב בו רשומות מסוימות תכלולנה נתונים לא אמינים, משום ששינוי רשומה מסוימת לא יוביל בהכרח לעדכון כל הרשומות בהן כלול הנתון ששונה.

שיפור ביצועי היישום כרוך בדרך כלל בוויתור על יעילות אחסון. עבור כל מסד נתונים שתעצב, תידרש למצוא מהו האיזון האופטימלי בין ביצועים ויעילות אחסון.

שימוש באינדקסים

בדרך כלל, מאוחסנות הרשומות במסד הנתונים על פי סדר הזנתן. סדר זה מכונה **סדר פיסי** (Physical Order) או **סדר טבעי** (Natural Order) של הרשומות. אולם, במקרים רבים תרצה לעיין בנתונים בסדר שונה מסדר הזנתם. כלומר, תרצה להגדיר עבורם **סדר לוגי** (Logical Order). תהליך איתור רשומה כלשהי בטבלה המסודרת בסדר פיסי עלול לגזול זמן רב.

אינדקס (Index) מקל על חיפוש נתונים בטבלה. אינדקס הוא טבלה מיוחדת של המערכת המכילה ערך מפתח (שנגזר לרוב מערך שדה מסוים או מספר שדות) עבור כל רשומה הכלולה בטבלה. האינדקס עצמו מאוחסן בסדר לוגי מיוחד ומכיל גם מצביעים המראים למנגנון מסד הנתונים היכן ממוקמת הרשומה הפיסית. אינדקס כזה מזכיר במידה מסוימת אינדקס של ספר. אינדקס של ספר מאפשר לך לאתר מילים ונושאים, מפני שהוא כולל מצביעים (מספרי עמודים), אשר מורים לך היכן תוכל למצוא את המידע הרלוונטי.

מדוע כדאי להשתמש באינדקס

מבנה האינדקס מאפשר להיעזר בו לביצוע פעולות מהירות לאיתור ואחזור מידע. אם הגדרת אינדקס אלפביתי עבור טבלה המכילה שמות, תוכל לאחזר רשומה המכילה שם מסוים באמצעות חיפוש באינדקס. כדי להבין את התועלת של שימוש באינדקס, נסה לתאר לעצמך ספר טלפונים, שהמנויים רשומים בו על-פי הסדר בו התחברו לחברת הטלפונים. אם זהו ספר טלפונים של עיר גדולה, איתור מספר מסוים עלול לדרוש זמן רב, מפני שתידרש לעיין בכל שורה ושורה בספר.

לטבלה מסוימת עשויים להיות מקושרים כמה אינדקסים, אשר יקושרו אליה כדי שניתן יהיה לארגן את המידע הכלול בה במספר אופנים. לדוגמה, את רשימת העובדים של החברה תוכל למיין על-פי שמות משפחה של העובדים, על-פי גיל העובדים, על-פי ותק בחברה או על-פי דרגות שכר. כל אינדקס יקל על חיפוש אותה סדרת נתונים.

אזהרה:

האפשרות לחפש מידע בחתכים שונים עשויה להיות מאוד רצויה עבורך, אך ניהול מספר אינדקסים במקביל עלול לפגום ברמת הביצועים, שכן כל שינוי החל בנתונים מחייב עדכון של כל אינדקס בנפרד. שוב תידרש לבצע ניתוח עלות-תועלת של פעולה באופנים שונים.

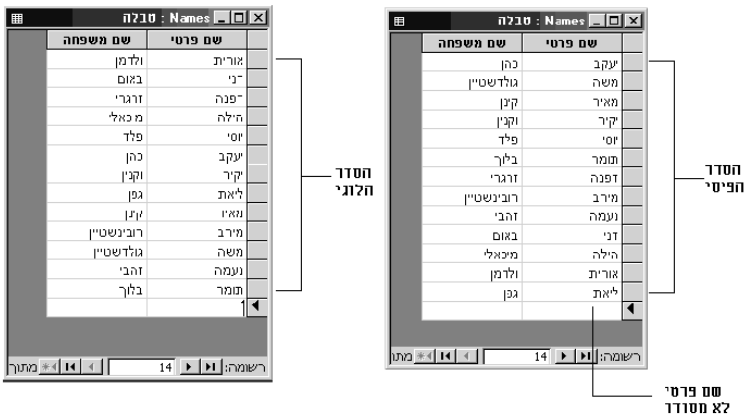
הערה:

תוכל להציג נתונים בטבלה בסדר שונה גם על ידי מיון שלהם, או על ידי שימוש במשפט SQL: ORDER BY. מנוע SQL אומנם אינו משתמש באינדקסים באופן ישיר, אך קיום אינדקסים מאיץ את הביצוע פעולות המיון אשר מבוצעות תוך שימוש במשפט ORDER BY. מידע נוסף על נושא זה תוכל למצוא בנספח 3 **תקציר פקודות SQL**.

ראה: "הגדרת תנאי המיון", נספח 3.

ביטויי מפתח יחיד

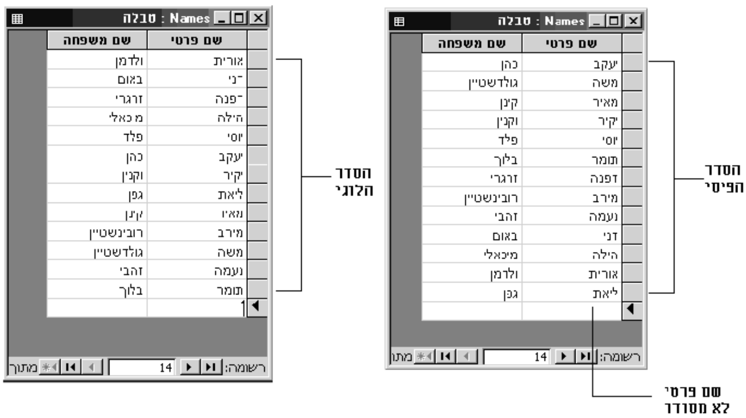
האינדקס הנפוץ ביותר הוא **אינדקס מפתח יחיד** (Single Key Index), אשר מתבסס על שדה יחיד בטבלה. דוגמאות אינדקסים כאלה הם מספר תעודת זהות, מיקוד, מספר עובד או שם משפחה. אם כלולות בטבלה מספר רשומות המכילות ערך זהה בשדה האינדקס, הן תוצגנה בסדר הפיסי שלהן בתצוגה שנבנתה על בסיס האינדקס. בתרשים 24.7 מוצגת הטבלה Names פעם, בצורתה המקורית, על פי הסדר הפיסי שלה ופעם כשהיא ממוינת על פי שדה Last Name.



תרשים 24.7: סדר פיסי וסדר לוגי עשויים להיות שונים. סדר לוגי תלוי באינדקס

ביטויי מפתחות מרובים

אינדקסים המבוססים על מפתח יחיד מאפשרים אומנם להציג מידע כשהוא מסודר בסדר מסוים, אך לעיתים נדרש למיין את מידע יותר בקפדנות. זאת ניתן לעשות על ידי שימוש ב**אינדקס מרובה מפתחות** (Multiple-Key Index). שם האינדקס מעיד על כך שמדובר באינדקס המבוסס על מספר שדות המאוחסנים בטבלה. השימוש בשם משפחה ובשם פרטי למיון רשימות אנשים הוא דוגמה שכיחה לשימוש באינדקס מרובה מפתחות. בתרשים 24.8 תוכל לראות כיצד השימוש בשם פרטי כבמפתח מיון, לצד שם משפחה, ישנה את הסדר בו תוצגנה הרשומות בטבלה. בדומה לאינדקסים המבוססים על מפתח יחיד, גם באינדקסים מרובי מפתחות, רשומות אשר ערכי האינדקסים שלהן זהים, תוצגנה בסדר הפיסי שלהן (כלומר סדר הזנתן לתוכנית).



תרשים 24.8: שימוש באינדקס מרובה מפתחות מעדן את הסדר הלוגי של הרשומות



תרשים 24.9: הגדרה לא נכונה של סדר השדות באינדקס, תוביל לתוצאות בלתי רצויות

אזהרה:

ייתכן שהדבר ייראה לך מובן מאליו, אך חשוב לחזור ולהזכיר שהסדר בו תגדיר את השדות באינדקס מרובה מפתחות ישפיע השפעה מכרעת על הסדר בו תוצגנה הרשומות. לדוגמה, מיון טבלה לפי שדה First Name תחילה, אחר לפי שדה Last Name יוביל לתוצאות שונות ממיון אותה טבלה לפי שדה Last Name תחילה, ואחר לפי שדה First Name. תרשים 24.9 מציג את התוצאות (הבלתי רצויות) של מיון הטבלה שהוצגה בתרשים 24,7 על פי אינדקס First Name/Last Name.

שימוש בשאילתות

בעת נרמול מידע, נהוג לאחסן פריטי מידע הקשורים זה לזה בטבלאות שונות. אך כאשר אתה נדרש לגשת למידע, אתה מעוניין לראות מידע המאוחסן בטבלאות שונות כאשר הוא מוצג במקום אחד. כדי שהדבר יהיה אפשרי, עליך להרכיב **מערך רשומות** (RecordSet) שיכלול בתוכו מידע הכלול בטבלאות שונות. אתה יכול ליצור מערך רשומות מטבלאות שונות על ידי שימוש במשפט SQL שבו יצוינו השדות הרצויים לך, מיקום אותם שדות והקשרים הקיימים בין הטבלאות. דרך אחת בה ניתן להשתמש במשפט SQL היא על ידי שילובו בשיטה OpenRecordset של האובייקט Database, אשר יוצרת את מערך הרשומות. אך תוכל לאחסן את משפט SQL גם כ**שאילתה** (Query) במסגרת מסד הנתונים עצמו.

לשימוש בשאילתות מאוחסנות יש מספר יתרונות:

- האחסון במסד הנתונים מקל על השימוש במשפט במספר מקומות בתוכנית שלך, או לחילופין בתוכניות שונות.
- קל יותר לבצע שינויים במשפט SQL המאוחסן במקום אחד.
- שאילתות מאוחסנות פועלות מהר יותר משאילתות הכלולות בקוד התוכנית.
- אחסון השאילתה במסגרת מסד הנתונים מקל על התאמת היישום שלך לפעולה בסביבת שרת/לקוח (Client/Server).

ראה: "תקציר פקודות SQL", נספח 3.

יישום התכנון

הצעד הראשון במימוש מסד הנתונים שתכננת הוא יצירת מסד הנתונים עצמו. אחד מהשיקולים אליהם אתה נידרש להתייחס הוא סוג מסד הנתונים בו תשתמש. סביבת מסד הנתונים הטבעית של Visual Basic היא Microsoft Jet Database Engine. מסדי נתונים מבוססי Jet מכונים במקרים רבים, מסדי נתונים מבוססי Access, זאת משום שתוכנת Access לניהול מסדי נתונים מבוססת על מנגנון Jet. ב- Visual Basic מוצעות לך ארבע שיטות שונות ליצירת מסדי נתונים מבוססי Jet. בעת בניית היישום שלך תוכל לבחור אחת מהן:

- יצירת מסד הנתונים על ידי תכנות, תוך שימוש ב**אובייקטי גישה לנתונים - DAO** (Data Access Objects).
- שימוש ביישום Visual Data Manager המסופק כחלק מ- Visual Basic.
- שימוש בתוכנת Access.
- שימוש בתוכנות לניהול מסדי נתונים מתוצרת יצרנים אחרים.

יצירת מסד נתונים באמצעות תכנות, תוך שימוש בכלי Data Access Objects מעניקה לתוכנית שלך שליטה מלאה על אופן היווצרותו ומבנהו של מסד הנתונים. נושא זה יידון בפרק 26 **שימוש באובייקטי גישה לנתונים - DAO**.

שאר השיטות ליצירת מסדי נתונים מפורטות להלן ותידונה בסעיפים הבאים:

- **Visual Data Manager**. השימוש בתוספת (Add In): Visual Data Manager, מאפשר לך ליצור מסדי נתונים וליצור, לשנות ולמחוק טבלאות, אינדקסים וקשרים בין טבלאות המוגדרים במסגרת מסדי נתונים.
- **Microsoft Access**. Access היא ככל הנראה הכלי השכיח ביותר ליצירת מסדי נתונים מבוססי Jet. יתרונה של Access הוא בכך שהיא מאפשרת לך להגדיר אינדקסים וקשרים בין טבלאות בשיטת **גרור ושחרר** (Drag And Drop).
- **תוכנות לניהול מסדי נתונים מתוצרת יצרנים אחרים**. כיום מוצעות תוכנות רבות לניהול מסדי נתונים, מסחריות או חופשיות, מבוססות Jet, או מבוססות נועים אחרים. חלק מהן ייעודיות ואחרות רב-תכליתיות, כגון Visual Data Manager.

Visual Data Manager

היישום Visual Data Manager, אשר מסופק יחד עם Visual Basic, מציע לך כלי אינטראקטיבי ליצירה ושינוי מסדי נתונים. את היישום ניתן להפעיל על ידי בחירה בפקודה **Visual Data Manager** מתפריט **Add-Ins** של Visual Basic.

הערה:

היישום Visual Data Manager מסוגל לנהל מסדי נתונים מהסוגים הבאים: Access (Jet), dBASE, FoxPro, Paradox, ODBC וקבצי טקסט. אך במסגרת יישומים הנכתבים ב-Visual Basic, הוא משמש לרוב לניהול מסדי נתונים מבוססי Access (Jet).

טיפ:

Visual Data Manager הוא גם אחד מיישומי הדוגמה של Visual Basic. עיון בפרויקט זה יילמד אותך דברים רבים על יצירת מסדי נתונים ב-Visual Basic.

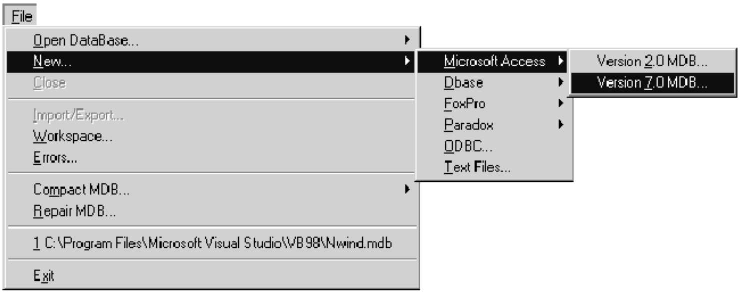
יצירת קובץ מסד הנתונים

יצירת קובץ מסד הנתונים (Database File) היא הצעד הראשון בתהליך יצירת מסד הנתונים. קובץ מסד הנתונים יהווה את המסגרת הפיסית שבתחומה תבוצענה שאר פעולות הבניה. כדי ליצור קובץ מסד נתונים באמצעות Visual Data Manager, בצע את הפעולות הבאות:

1. הפעל את הפקודה **New** מתפריט **File**. הפקודה תציג לפניך תפריט משנה ממנו תוכל לבחור את סוג הקובץ שתיצור.
2. לצורך דוגמה זו, צור מסד נתונים מסוג Access (Jet), על ידי בחירת האפשרות **Microsoft Access** מהתפריט. פעולה זו תגרום לפתיחת תפריט משנה המאפשר לך לבחור את גרסת Jet בה ישתמש מסד הנתונים שתיצור.
3. אם אתה מתכוון לחלוק נתונים עם משתמשי Windows 3.x, בחר בגירסה 2.0, אחרת בחר בגירסה 7.0. תרשים 24.10 מציג את התפריט הראשי ותפריטי המשנה בהם תשתמש כדי ליצור מסד נתונים.

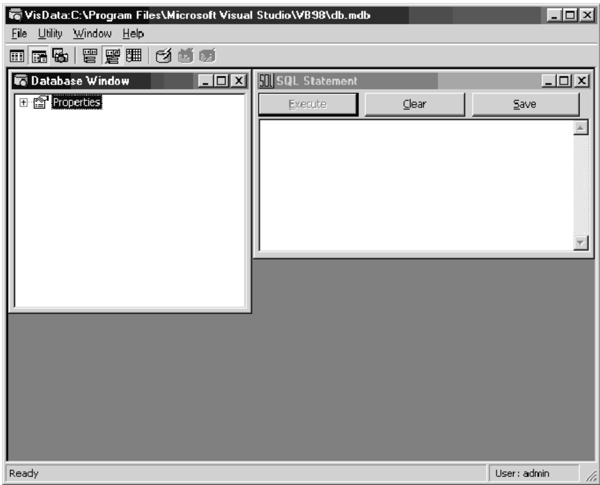
לאחר בחירת סוג מסד הנתונים שלך, תוצג בפניך תיבת הדו-שיח **Select Microsoft Access Database to Create**. תיבה דו-שיח זו מאפשרת לך לבחור שם ומיקום (תיקיית אחסון) עבור מסד הנתונים.

4. הגדר שם עבור מסד הנתונים ובחר את התיקיה בה אתה מעוניין לאחסנו. לאחר מכן, לחץ על לחצן **Save**. פעולה זו תעביר אותך למצב עיצוב (Design Mode), המתואר בתרשים 24.11.



תרשים 24.10: התפריטים השונים מאפשרים לבחור סוג וגירסה עבור מסד הנתונים שלך

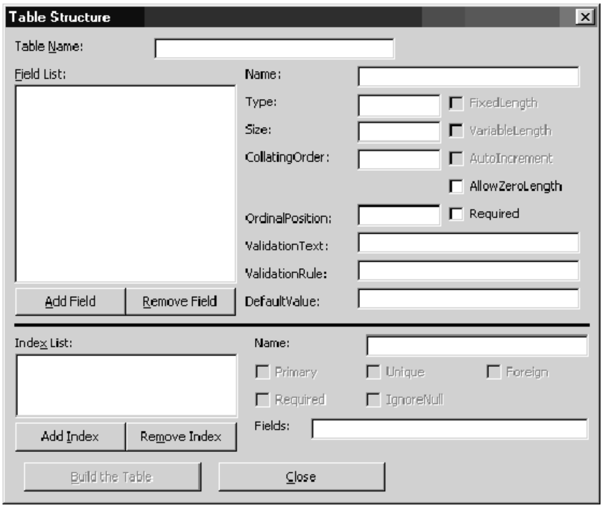
Visual Data Manager מציג את המידע הכלול במסד הנתונים במבנה עץ. תצוגה זו מאפשרת לצפות בנוחות בטבלאות ושאילתות הכלולות במסד הנתונים. צורה זו של הצגת המידע מאפשרת גם לצפות בנתונים ברמת פירוט גבוהה יותר, תוך הצגת שדות ואינדקסים. אם תרצה, תוכל להציג עד לרמת הפירוט הנמוכה ביותר כדי לראות מאפייני שדה.



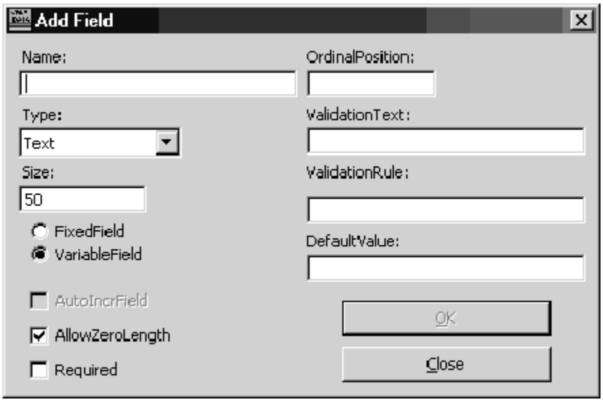
תרשים 24.11: חלון Database של היישום Visual Data Manager מאפשר לך גישה אל פונקציות לעיצוב טבלאות, שדות ואינדקסים

הוספת טבלה חדשה

לאחר יצירת מסד הנתונים, תוכל ליצור במסגרתו טבלאות. ליצירת טבלה חדשה, לחץ לחיצה ימנית על נקודה כלשהי בחלון **Database** ובחר מהתפריט שיוצג את הפקודה **New Item**, כדי להציג את תיבת הדו-שיח **Table Structure** אשר מוצגת בתרשים 24.12. תיבת דו-שיח זו תציג בפניך מידע אודות הטבלה עצמה וכן אודות השדות והאינדקסים המוגדרים במסגרתה. תיבת הדו-שיח כוללת גם לחצנים המאפשרים לך להוסיף או לגרוע שדות ואינדקסים. כדי להוסיף שדה לטבלה, לחץ על הלחצן **Add Field**. פעולה זו תגרום להצגת תיבת הדו-שיח **Add Field** המוצגת בתרשים 24.13.



תרשים 24.12: תיבת הדו-שיח Table Structure מאפשרת לך להגדיר שם עבור טבלה



תרשים 24.13: בתיבת הדו-שיח Add Field תוכל להגדיר מאפייני שדות בטבלה

כדי להמשיך בבניית מסד הנתונים לדוגמה שלנו, הגדר תחילה שם עבור הטבלה בתיבת הטקסט **Table Name** ולאחר מכן בצע את הפעולות הבאות עבור כל שדה שתרצה להוסיף:

1. לחץ על הלחצן **Add Field** בתיבת הדו-שיח **Table Structure**.
2. בחר את סוג הנתון שיאוחסן בשדה מהרשימה הנפתחת **Type**.
3. הגדר את גודל השדה (אם תידרש לעשות זאת).
4. הגדר פרמטרים אופציונליים כדוגמת חוקי אימות.
5. לחץ על הלחצן **OK** כדי לצרף את השדה לטבלה.

לאחר שתוסיף את כל השדות הדרושים לטבלה שלך, לחץ על הלחצן **Close** בתיבת הדו-שיח **Add Field**, כדי לשוב אל תיבת הדו-שיח **Table Structure**.

אם תרצה לגרוע שדה מהטבלה, בחר את שם השדה מרשימת השדות של תיבת הדו-שיח ולחץ על הלחצן **Remove Field**. כאשר תהיה מרוצה מהשדות הכלולים בטבלה, לחץ על הלחצן **Build The Table**, כדי ליצור את הטבלה.

שינוי שדות

לאחר שתגדיר שדות בטבלה, תוכל לשנות את מאפייניהם באמצעות תיבת הדו-שיח Table Structure הנגישה בלחיצה ימנית על שדה ובחירת הפקודה Design מהתפריט המקוצר שיופיע. לשינוי המאפיינים, בחר את שם השדה מרשימת השדות. מאפייני השדה, אותם תוכל לשנות, יוצגו בתיבת הדו-שיח בצורת טקסט פעיל (Enabled) או תיבות סימון פעילות. שאר המאפיינים יוצגו בצורת פקדים מנוטרלים.

טיפ:

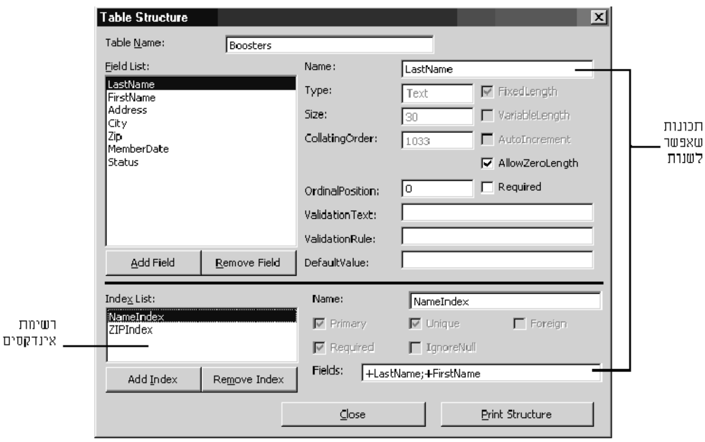
את הגדרות מאפייני השדות ניתן לערוך גם מהחלון Database של Visual Data Manager. תידרש להרחיב את תצוגת מסד הנתונים, כך שמאפייני השדות יוצגו, וללחוץ לחיצה ימנית על המאפיין שאת ההגדרה שלו אתה מעוניין לשנות. תוכל לבחור את הפקודה Edit מהתפריט שיוצג כדי לשנות את הגדרת המאפיין.

הערה:

ב- Visual Basic לא ניתן לערוך או למחוק שדה שהוא חלק מאינדקס או מקשר. אם תרצה למחוק שדה כזה, יהיה עליך קודם למחוק את האינדקס או את הקשר ורק אחר כך תוכל לבצע שינויים בשדה.

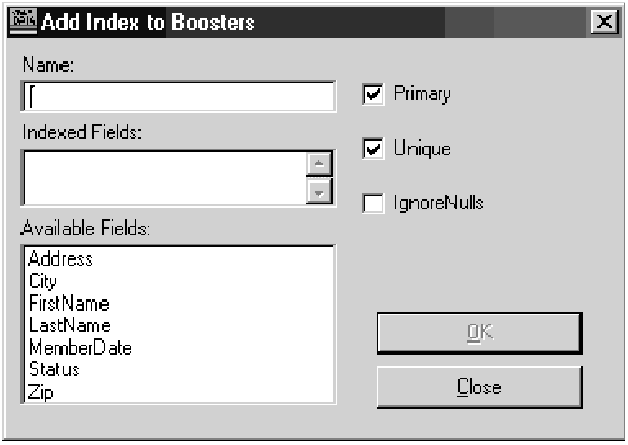
הוספת אינדקס לטבלה

בתיבת הדו-שיח Table Structure תוכל גם להוסיף, לשנות ולגרוע אינדקסים המוגדרים בטבלה. כל האינדקסים המוגדרים מוצגים בתיבת הרשימה **Index List**, המוצגת בחלק התחתון של תיבת הדו-שיח, כמתואר בתרשים 24.14.



תרשים 24.14: ניתן לערוך ולגרוע אינדקסים על ידי תיבת הדו-שיח Table Structure

להוספת אינדקס חדש, לחץ על הלחצן Add Index. תוצג תיבת הדו-שיח Add Index המוצגת בתרשים 24.15. בתיבה זו, הגדר תחילה שם עבור האינדקס ואחר בחר את השדות שייכללו באינדקס, על ידי לחיצה עליהם בתיבה Field List. כל שדה שתלחץ עליו יצורף לרשימת השדות הכלולים באינדקס, שהשדות יוצגו בה לפי הסדר בו נבחרו. כברירת מחדל, כל השדות ממוינים בסדר עולה. לשינוי סדר המיון של שדה לסדר יורד, הוסף סימן מינוס (-) לפני שם השדה ברשימת השדות הכלולים באינדקס.



תרשים 24.15: תיבת הדו-שיח Add Index מהווה עזר **ויזואלי** להגדרת אינדקסים

לאחר שהגדרת את השדות הנכללים באינדקס, תוכל להגדיר את האינדקס כ**אינדקס ייחודי** (Unique Index) או כ**אינדקס ראשי** (Primary Index) - אך זאת רק אם טרם הוגדר אינדקס ראשי עבור אותה טבלה. פעולה זו מתבצעת על ידי בחירת תיבת הסימון המתאימה מתוך תיבת הדו-שיח. כאשר תסיים להגדיר את האינדקס על-פי הדרישות שלך, שמור אותו על ידי לחיצה על הלחצן OK. האינדקס שיצרת זה עתה יתווסף לרשימת האינדקסים המוצגת בתיבת הדו-שיח Table Structure. כל שיהיה עליך לעשות כדי לגרוע אינדקס הוא לבחור את הפריט שלו ברשימת האינדקסים וללחוץ על הלחצן **Remove Index**.

חזרה לחלון העיצוב של Visual Basic

סגירת החלון של Visual Data Manager או לחילופין בחירה בפקודה **Exit** מתפריט **File** תחזרנה אותך אל חלון העיצוב של Visual Basic (ניתן גם לעבור הלוך ושוב בין Visual Data Manager לבין סביבת הפיתוח של Visual Basic). משום ש-Visual Data Manager הוא יישום שלם, ייתכן שתרצה להדר אותו ולהוסיף קיצור דרך לקובץ EXE שלו בשולחן העבודה או לתפריט התחלה. פעולה זו תחסוך את הצורך לטעון את Visual Basic בכל פעם שתרצה להשתמש ב-Visual Data Manager.

הצגה ושינוי מבני טבלאות

לאחר שיצרת טבלה, תוכל להציג את המבנה שלה ואף לשנות פרטים מסוימים במבנה. לחץ לחיצה ימנית על שם הטבלה בחלון Database ובחר את הפקודה Design מהתפריט שיוצג לפניך. תראה כי על המסך מופיעה תיבת הדו-שיח Table Structure, עמה עבדת בסעיפים הקודמים. תיבה זו תאפשר לך לעיין במבני השדות והאינדקסים אשר מרכיבים את הטבלה.

לאחר יצירת הטבלה, אין אפשרות לשנות את סוג המידע המאוחסן בשדה או את גודל השדה מתוך Visual Data Manager. אולם עדיין מצויה בידך היכולת להוסיף, לגרוע או לשנות שדות בטבלה. תוכל גם לשנות מאפיינים מסוימים של שדות. בדומה לשדות, לא תוכל לשנות גם את מבנה האינדקסים המוגדרים בטבלה מתוך Visual Data Manager, אך תוכל להוסיף, לגרוע, ולשנות שמות אינדקסים.

שינוי או מחיקת טבלאות

לחיצה ימנית על שם טבלה בחלון Database מאפשרת לשנות שם או למחוק את הטבלה.

בחירה בפקודה Rename מהתפריט המקוצר תגרום להצגת שם הטבלה בתיבת עריכה. תוכל פשוט להקליד את השם החדש בתיבה ולהקיש Enter. אם תחליט שלא לשנות את שם, הקש Esc.

בחירה בפקודה Delete תגרום להצגת תיבת דו-שיח שבה תתבקש לאשר את מחיקת הטבלה. לחיצה על Yes, תגרום למחיקת הטבלה ממסד הנתונים. פעולה זו היא בלתי הפיכה.

העתקת טבלה

אם תרצה ליצור עותק של טבלה, בחר את הפקודה Copy Structure מהתפריט המקוצר שיוצג לפניך בעקבות לחיצה ימנית על שם הטבלה בחלון Database.

בחר את שם הטבלה שברצונך להעתיק מתיבת הדו-שיח Copy Structure. שם מסד הנתונים הפעיל יוצג בתיבה Target Database. אם תרצה להעתיק את הטבלה אל מסד נתונים אחר יהיה עליך להקליד את שמו בתיבה זו. כברירת מחדל, תיבת הסימון Copy Indexes מסומנת. אם אינך מעוניין שהטבלה החדשה תכיל אותם אינדקסים שהגדרת עבור הטבלה המקורית, תוכל לבטל את סימון התיבה. תיבת הסימון Copy Data, שכברירת מחדל אינה מסומנת, מאפשרת לך להעתיק את הרשומות הכלולות בטבלה אל הטבלה החדשה. אם תותיר את תיבת הסימון הזו כשהיא בלתי מסומנת, לטבלה החדשה יהיה מבנה זהה לזה של הטבלה המקורית אך היא לא תכיל נתונים כלשהם. לאחר שתבחר את האפשרויות הרצויות לך, לחץ על OK כדי ליצור את הטבלה החדשה.

יצירת מסד נתונים בעזרת כלים אחרים

Visual Data Manager אינו כמובן האמצעי היחיד ליצירת מסדי נתונים. הוא בסך הכל יישום לדוגמה המסופק יחד עם Visual Basic אשר מבצע עבורך פעולה מועילה מאוד. אך כיום מוצעים לך כלים נוספים רבים המאפשרים לך ולמשתמשי היישומים שלך ליצור ולנהל מסדי נתונים.

שימוש ב-Access

Microsoft Access היא אחת האפשרויות המוצעות לך ככלי ליצירת מסדי נתונים מבוססי Jet. Access מהווה ממשק ויזואלי נוח ליצירת טבלאות, אינדקסים, שאילתות וקשרים בין טבלאות. אך זוהי אפשרות העומדת לרשותך רק אם רכשת עותק של התוכנה. Visual Basic מסוגלת לעבוד עם כל גרסאות Access, אך אם תרצה לנצל יכולות של ניהול מסדי נתונים בני 32 סיביות, תידרש להשתמש ב-Access 95 או בגירסה חדישה יותר.

תוכנות לעיצוב מסדי נתונים מתוצרת יצרנים אחרים

בנוסף ל-Visual Data Manager ו-Access גם יצרנים רבים אחרים מציעים תוכנות אשר תאפשרנה ליצור ולנהל מסדי נתונים מבוססי Jet. חלק מתוכנות אלו מציע יכולות מתקדמות בתחומי עיצוב המידע. יכולות אלו תסייענה לך לקבוע אילו נתונים ייכללו באילו טבלאות ולהגדיר את מערכת הקשרים בין הטבלאות. לאחר שתסיים לעצב את מסד הנתונים, התוכנה תחולל את המסד עבורך באופן אוטומטי.

אך כדאי להיות זהיר במה שקשור לבחירת תוכנות מתוצרת יצרנים אחרים. חלקן מיועדות למטרות ממוקדות מאוד ואילו אחרות רב-גוניות יותר. Access חזקה יותר מרוב התוכנות הרב-תכליתיות לניהול מסדי נתונים המוצעות על ידי יצרנים אחרים, אך אם יש לך צורך ממוקד אשר מטופל בצורה טובה יותר על ידי תוכנה אחרת, אל תפסול אותה על הסף. בטרם תרכוש כלי, כדאי לך לבחור כלי שהמפיץ שלו יאפשר לך לבחון אותו בטרם תרכוש אותו.

יתרונות התוכנה הייעודית

בפרק זה למדת כי Visual Data Manager ו-Access יכולים ליצור, לשנות ולמלא מסדי נתונים בנתונים. לכן, ייתכן שתשאל את עצמך, "מדוע בכלל כדאי לי להשקיע מאמץ בתכנות היישום שלי"? התשובה לשאלה זו היא שבמקרים רבים כלל לא תידרש להשקיע מאמץ כזה. אם יש לך שליטה ישירה על מסד הנתונים (כלומר אם אתה המשתמש היחיד של המסד או שאתה יכול לגשת אליו בכל עת), ייתכן שכלל לא תידרש להשתמש בפקודות כדי לבצע בו שינויים.

אולם, אם היישום שלך מופץ למשתמשים רבים, במסגרת ארגון מסוים או בתפוצה רחבה יותר - הרי שהשימוש בתוכנית לניהול מסד הנתונים מציע לך מספר יתרונות. אחד היתרונות הוא בתחום אתחול ראשוני של מסד הנתונים. אם השגרות לאתחול

מסד הנתונים הן חלק מהתוכנית, לא תידרש לכלול עותק ריק של מסד הנתונים במסגרת עותקי ההפצה של התוכנית. השמטת העותק הריק של מסד הנתונים תסייע לצמצם את מספר הדיסקטים או התקליטורים שיידרשו לצורך הפצת היישום, פעולה אשר תקטין את הסיכוי להשמטת קבצים חיוניים מעותקי ההפצה.

המשתמש גם עלול למחוק בטעות את קובץ מסד הנתונים, מצב אשר יאלץ אותו ליצור קובץ חדש.

יתרון נוסף של שימוש בתוכנית, קשור בתחום הפצת עדכונים עבור היישום. הכללת השינויים בתוכנית תאפשר למשתמש פשוט להריץ את התוכנית כדי לבצע את העדכונים הדרושים בתוכנה או במבנה מסד הנתונים. המשתמש לא יידרש לטעון נתונים אל תוך קובץ ריק. ביצוע העדכון באתר הלקוח יאפשר לך גם לעקוב אחר שינויים במבנה מסד הנתונים שבוצעו על ידי משתמשי קצה.

שיקולים הקשורים בביצועים הם סיבה נוספת המעודדת הכללת פקודות ליצירת מסד הנתונים ולתחזוקתו במסגרת התוכנית. מבחינת ביצועים, רצוי לעיתים ליצור טבלה זמנית לשם האצת פעולת התוכנית או לשם אחסון תוצאות ביניים. טבלה אשר תימחק עם סיום פעולת התוכנית. ייתכן גם שתרצה ליצור אינדקס זמני, לשם הגדרת סוג מיון מסוים או לשם האצת פעולות חיפוש.

מכאן...

בפרק זה למדת אודות עיצוב ויצירה של מסדי נתונים שישמשו במסגרת יישומים. אולם, כדי שיתאפשר לך להשתמש במסד הנתונים, תידרש לכתוב **יישום לגישה אל מסד הנתונים** (Database Access Application). בנושא זה נעסוק בפרקים הבאים. מידע נוסף תוכל למצוא בפרקים המפורטים להלן:

- מידע אודות השימוש בכלי Data Control (פקד הנתונים) של Visual Basic בשילוב עם פקדים אחרים, לשם יצירה מהירה של יישומים שיעבדו עם מסדי נתונים, תוכל למצוא בפרק 25 **פקד הנתונים ופקדי איגוד נתונים**.
- מידע על שיטות תכנות מתקדמות יותר תוכל למצוא בפרק 26 **שימוש באובייקטי גישה לנתונים (DAO)**, ובפרק 27 **גישה לאובייקטי נתונים מרוחקים (RDO)**.
- דיון בשיטה החדישה ביותר לעבודה עם מסדי נתונים, המוצעת כיום על ידי Microsoft יובא בפרק 28 **גישה לאובייקטי נתונים באמצעות פקדי ActiveX**.

25

פקד הנתונים ופקדי איגוד נתונים

מה בפרק?

- הבנת פקד הנתונים
- היכרות עם פקדי איגוד נתונים
- יצירת יישום פשוט
- יצירה אוטומטית של טפסים

Visual Basic מיועדת לאפשר למפתחים ליצור יישומים המיועדים לסביבת Windows, במהירות ובקלות. פשטות השימוש בסביבת הפיתוח תקפה גם בעת פיתוח יישומים לניהול מסדי נתונים. אם יש לך מסד נתונים קיים שאתה מעוניין להשתמש בו, Visual Basic מאפשרת לך לפתח יישום שלם לניהול מידע, תוך השקעה מינימלית בתכנות. עליך רק לשמוט מספר פקדים על הטופס ולהגדיר מאפיינים אחדים עבורם. Visual Basic מקלה עליך את הפיתוח גם בכך שהיא יוצרת עבורך טפסי הזנת מידע, באופן אוטומטי.

מרכיבי סביבת הפיתוח, המאפשרים לך ליהנות מיכולות אלו, הם **פקד הנתונים** (Data Control) **ופקדי איגוד נתונים** (Data-Bound Controls). כלים אלה מאפשרים לך ליצור מיגוון רחב של יישומים. אך בטרם תפתח צפיות גבוהות מדי, עליך לדעת שככל שהיישומים שתפתח יהיו מתקדמים יותר, כך יגדל נפח התכנות שתצטרך לבצע בעצמך ותקטן מידת הסיוע שיוכלו להעניק לך כלי הפיתוח של Visual Basic. אולם, אם הינך חדש בתחום התכנות ב- Visual Basic, כלים אלה יהוו עבורך נקודת התחלה טובה בפיתוח יישומים לעבודה מול מסדי נתונים ויאפשרו לך ליצור אב-טיפוסים של יישומים במהירות ובקלות.

פקד הנתונים

פקד הנתונים הוא אמצעי הסיוע העיקרי שמציעה לך Visual Basic לפיתוח יישומים המיועדים לעבוד מול מסדי נתונים. פקד הנתונים הוא אחד הפקדים הכלולים ב**ארגז הכלים** (Toolbox) של Visual Basic. כדי להוסיף פקד נתונים ליישום שלך, עליך לבצע ארבע פעולות פשוטות:

1. בחר את פקד הנתונים בארגז הכלים.
2. שרטט את הפקד על הטופס.
3. הגדר את מאפיין DatabaseName של הפקד.
4. הגדר את מאפיין RecordSource של הפקד.

הערה:

ארבע הפעולות אלו הן המינימום הנדרש לשם הוספת פקד הנתונים ליישום המבצע גישה למסד נתונים. מנגנון Jet מאפשר לך להשתמש במספר סוגים של מסדי נתונים (הסוגים הנתמכים על ידי המנגנון מפורטים במסגרת מאפיין Connect). אם תרצה לבצע גישה אל מסדי נתונים שאינם נתמכים על ידי Jet, כגון SQL Server, תידרש להגדיר מאפיינים נוספים.

ראה: פקד נוסף לניהול מידע, הפקד ActiveX Data Control, מתואר בפרק 28, **"גישה לאובייקטי נתונים באמצעות פקדי ActiveX"**.

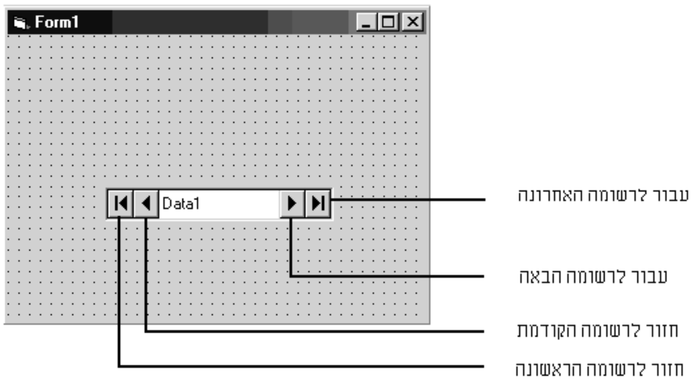
מהו פקד הנתונים

פקד הנתונים מהווה, למעשה, קישור בין נתונים שבמסד הנתונים שלך, לבין **פקדי איגוד נתונים** (Data Bound Controls) אשר משמשים להצגת הנתונים. בעת הגדרת מאפייני פקד הנתונים, הינך מורה לפקד לאיזה מסד נתונים עליו להתקשר ולאיזה חלק של אותו מסד נתונים עליו ליצור גישה. פקד הנתונים מעביר את הנתונים אל התוכנית שלך בצורת **מערך רשומות** (Recordset). מערך רשומות הוא אוסף רשומות במסד הנתונים. כברירת מחדל, פקד הנתונים יוצר מערכי רשומות מסוג Dynaset (קבוצת רשומות דינמית). מערכים אלה מכילים נתונים המאוחסנים באחת או יותר מהטבלאות במסד הנתונים שלך.

הערה:

ניתן להשתמש במספר סוגים של מערכי רשומות. Dynaset (מערך רשומות דינמי) הוא אוסף רשומות המשתנה באופן דינמי עם כל שינוי בנתונים שבמסד הנתונים עליו הוא מבוסס. סוג נוסף הוא מערך רשומות מסוג Snapshot ("תצלום"), המכיל תמונה סטטית של הנתונים, כלומר, מציג את הנתונים בהתאם לתמונת המצב בזמן בניית המערך, מבלי לשקף שינויים המתחוללים במסד הנתונים באופן שוטף.

בנוסף להפיכת הנתונים זמינים עבור התוכנית שלך, פקד הנתונים גם מקנה לך יכולות לניווט בין רשומות. הלחצנים המוצגים בתרשים 25.1 מאפשרים למשתמש לעבור לרשומה ראשונה במערך הרשומות, לרשומה אחרונה, לרשומה קודמת, או לרשומה הבאה. הצורה בה מעוצבים הלחצנים מבהירה יפה את ייעודם והם דומים ללחצנים שבמכשירים להשמעת תקליטורים או ללחצנים במכשירי וידאו.



תרשים 25.1: לחצנים הדומים לאלה שבמכשירי וידאו מציעים למשתמשים בפקד הנתונים יכולות לניווט בין רשומות

מערך הרשומות, אותו ייצור פקד הנתונים, ייקבע על סמך הגדרות המאפיינים DataBase ו-RecordSource של הפקד. אם הגדרת את המאפיינים האלה בעת עיצוב היישום, מערך הרשומות ייווצר במהלך טעינת הטופס בו כלול פקד הנתונים. ברוב המקרים, מערך הרשומות יישאר פעיל עד למחיקת הטופס מן הזיכרון ואז ייעלם.

הערה:

מערך הרשומות אינו מכיל נתונים, אלא מהווה אובייקט לוגי המייצג (או מצביע על) נתונים שבמסד נתונים פיסי. לאחר היעלמות מערך הרשומות או סגירתו, הנתונים הפיסיים נותרים ללא שינוי בטבלאות שבמסד הנתונים אליו התייחס המערך.

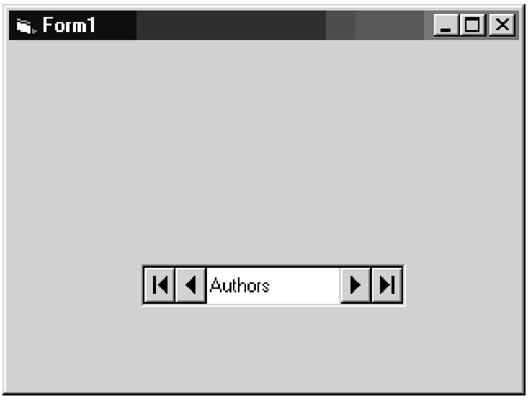
הוספת פקד נתונים לטופס

הצעד הראשון בשימוש בפקד הנתונים הוא הוספתו לטופס היישום. בחר את פקד הנתונים בארגז הכלים. מקם אותו במקום הרצוי לך על הטופס והגדר עבורו את הגודל הרצוי. לאחר שתמקם אותו ותגדיר את גודלו, תוכל להגדיר את המאפיינים Name ו-Caption.

מאפיין Name מגדיר את שם הפקד, אשר ישמש אחר כך כמזהה הפקד מול פקדי איגוד נתונים. שם ברירת המחדל אותו מקצה Visual Basic לפקד הנתונים הראשון שמתווסף לטופס, הוא Data1. אם ברצונך להעניק לפקד שם אחר, סמן את מאפיין Name בחלון המאפיינים של הפקד והקלד את השם הרצוי לך.

מאפיין Caption מגדיר את הטקסט שיוצג על הפקד. סביר להניח שתרצה להתאים את הכיתוב שעל הפקד לנתונים שאליהם הוא מספק גישה. ערך ברירת המחדל של מאפיין Caption זהה לערך ברירת המחדל של מאפיין Name. את הגדרת המאפיין ניתן לשנות באותו אופן בו שינית את הגדרת מאפיין Name.

פרק זה יערוך דיון בפקד הנתונים תוך היצמדות ליישום דוגמה, דבר אשר יאפשר לבצע פעולות שונות תוך כדי מהלך הלימוד. ניצור יישום פשוט אשר יציג את שמות הסופרים שבקובץ מסד הנתונים BIBLIO.MDB, המצורף אל Visual Basic כמסד נתונים לדוגמה. התחל את פרויקט הדוגמה בתור פרויקט מסוג Standard EXE והוסף פקד נתונים לטופס. הגדר את מאפיין Name כ-dtaMain ואת מאפיין Caption כ-Authors. ודא גם כי גודל הפקד מספיק כדי להציג את הכיתוב ללא קטיעה. תרשים 25.2 מציג את הטופס לאחר הוספת הפקד.



תרשים 25.2: שרטט את פקד הנתונים על הטופס והגדר עבורו כיתוב מתאים

טיפ:

ניתן גם לכלול בתוכנית שלך קוד אשר יגרום לכיתוב על פקד הנתונים להשתנות בהתאם לנתון המאוחסן ברשומה הפעילה, כגון שם אדם.

שני המאפיינים ההכרחיים

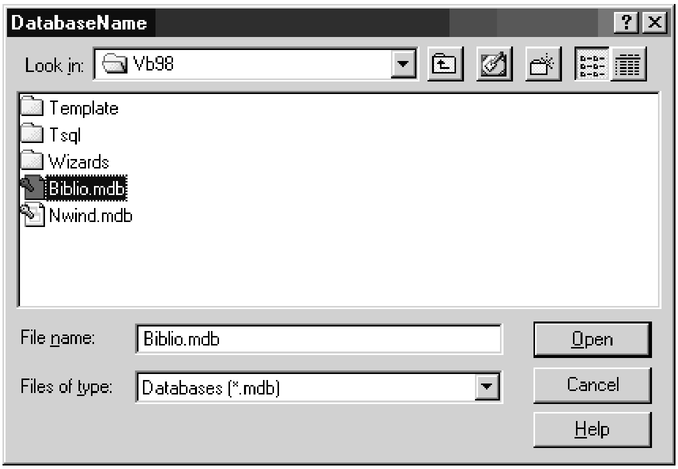
לאחר שתמקם את פקד הנתונים על הטופס, תידרש להגדיר קישור בינו לבין מסד הנתונים. את הקישור תגדיר על ידי הצבת ערכים לכמה ממאפייני הפקד. אומנם, קיימים מספר מאפיינים המשפיעים על אופן הקישור של פקד הנתונים למסדי נתונים, עם זאת, כשמדובר במסדי נתונים מבוססי Jet, עליך להתייחס לשני מאפיינים בלבד: DatabaseName ו-RecordSource. הצבת ערכים במאפיינים אלה מורה לפקד לאיזה מסד נתונים עליו להתקשר וגורמת לו להגדיר מערך רשומות המאפשר גישה לנתונים, לקריאה ולכתיבה.

הערה:

מאפיין DatabaseName שונה ממאפיין Name, אותו היכרת זה מכבר. מאפיין Name מגדיר את שם הפקד ומשמש לזיהויו במסגרת הקוד. מאפיין DataBaseName, לעומת זאת, מגדיר את הנתיב למסד הנתונים הפיסי, אליו ניתן לגשת באמצעות הפקד.

בחירת מסד נתונים

פקדי Data ניתנים לשימוש בשילוב עם מיגוון רחב של מסדי נתונים מסוגים שונים. אולם, אנו נתמקד בשילוב פקדים אלה בעבודה מול מסדי נתונים מבוססי Jet בלבד. בעת עבודה עם מסדי נתונים מסוג זה, מאפיין DatabaseName מכיל את שם קובץ מסד הנתונים. להצבת שם קובץ במאפיין, בחר את מאפיין DatabaseName בחלון המאפיינים של הפקד והקלד את נתיב הגישה המפורט אל הקובץ.



תרשים 25.3: תוכל להזין שם קובץ מסד נתונים בשורת המאפיין DatabaseName או על ידי בחירת הקובץ מתיבת הדו-שיח DatabaseName

הדרך הפשוטה ביותר לאיתור הקובץ היא על ידי חיפושו באופן ידני. לאיתור הקובץ, לחץ על הלחצן עליו מופיעות שלוש נקודות (...) הממוקם מימין לשורת המאפיין DatabaseName. פעולה זו גורמת להצגת תיבת הדו-שיח DatabaseName, המוצגת בתרשים 25.3. מצא את הקובץ הרצוי ולחץ על OK. הנתיב ושם הקובץ יוצבו במאפיין DatabaseName באופן אוטומטי.

אזהרה:

חיפוש קובץ באמצעות תיבת הדו-שיח הנפתחת משורת המאפיין, תציב בשורה נתיב מפורט לקובץ, לדוגמה, C:\MyData\LMS\TMS1112.MDB. הצבת ערך כזה עלולה לגרום לכך שבזמן ריצה התוכנית תחפש את מסד הנתונים באותו מקום בדיסק. כדאי שתאפשר גמישות מסוימת בהגדרת המקום בו מאוחסן הקובץ. מסיבה זו כדאי שתגדיר את מאפיין DatabaseName מבלי להשתמש בשם נתיב (לדוגמה, TMS112.MDB), מצב אשר יגרום לתוכנית לחפש את הקובץ בתיקיה הפעילה. אחר תוכל להגדיר נתיב יחסי תוך התייחסות לתיקיה הפעילה (\LMS\TMS1112.MDB). תוכל גם להגדיר את המאפיין מקוד התוכנית, תוך שימוש בקלט או בפרמטרים המוגדרים בעת אתחול התוכנית, כמו בדוגמה הבאה:

```
Dim sDBLocation As String
sDBLocation = App.Path & "\MyDB.MDB"
Data1.DatabaseName = sDBLocation
```

בחירת מערך רשומות

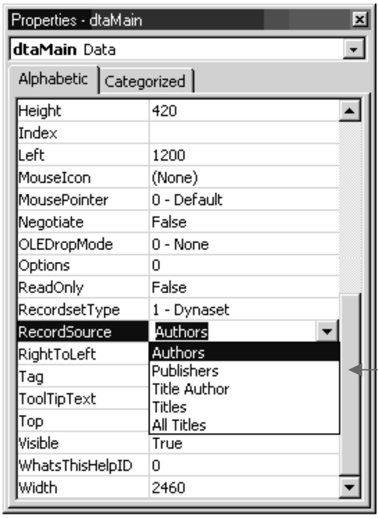
בתום הגדרת מאפיין DatabaseName, תוכל להגדיר את המידע אשר אתה מעוניין לאחזר ממסד הנתונים, תוך שימוש במאפיין RecordSource. אם אתה מעוניין כי הפקד יעבוד מול טבלה אחת במסד הנתונים, תוכל להציב במאפיין את שם הטבלה או לבחור את הטבלה מרשימת הטבלאות המוצגת בתרשים 25.4.

אם תרצה לעבוד רק עם נתונים מסוימים מתוך טבלה אחת או מספר טבלאות, תוכל להציב במאפיין RecordSource משפט SQL (Structured Query Language - שפת שאילתות מובנית). כדי לעשות זאת, תוכל להציב במאפיין אובייקט QueryDef הכלול במסד הנתונים ומכיל את משפט SQL או לחילופין, להציב משפט SQL חוקי. תוכל להשתמש בכל משפט SQL המגדיר מערך רשומות (תוכל אף לכלול במשפט כזה פונקציות משלך). אם החלטת להשתמש באובייקט QueryDef, עליך לוודא כי הוא אכן מוגדר ומאוחסן במסד הנתונים.

טיפ:

אם תרצה לוודא שמשפטי SQL אכן פועלים כנדרש, תוכל לבדוק אותם על ידי שימוש ב- Visual Data Manager ולהציב אותם במאפיין RecordSource על ידי חיתוך והדבקה.

ראה: "החלקים השונים של משפטי SQL", נספח 3.



תרשים 25.4: תוכל לבחור את הערך שיוצב במאפיין RecordSource מתוך רשימת טבלאות ואובייקטי QueryDef המוגדרים במסד הנתונים

היכרות עם פקד איגוד נתונים

בזאת השלמת את תהליך קישור הפקד למסד הנתונים והגדרת מערך רשומות שישמש אותו. אולם, כדי שתוכל להשתמש בנתונים, עליך לבצע פעולה נוספת: הגדרת **פקדי איגוד נתונים** (Data-Bound Controls). ב- Visual Basic, פקדי איגוד נתונים הם פקדים המוגדרים באופן הגורם להם לפעול בשילוב עם פקד הנתונים לשם יצירת יישומים של ניהול מסדי נתונים. לפיכך, ניתן לומר שהפקדים "מקושרים" לנתונים המאוחסנים במסד הנתונים. רוב פקדי איגוד הנתונים הם פקדים רגילים, הכוללים מספר מאפיינים נוספים המאפשרים להם לבצע פעולות גישה לנתונים. חלק מה**פקדים המותאמים אישית** (Custom Controls) של Visual Basic מיועדים במיוחד לשם עבודה עם פקדי Data.

חלק מהפקדים העשויים לשמש כפקדי איגוד נתונים כבר מוכרים לך:

	TextBox - תיבת טקסט
	Label - תווית
	CheckBox - תיבת סימון
	PictureBox - תיבת תמונה
	Image - תמונה

מה עושים הפקדים האלה

כל פקד איגוד נתונים מקושר לפקד הנתונים או יותר נכון, לשדה מסוים במערך הרשומות המקושר לפקד. פקד איגוד הנתונים מציג באופן אוטומטי את הנתונים המאוחסנים באותו שדה ברשומה הפעילה. כאשר המשתמש עובר לרשומה אחרת, על ידי שימוש בלחצני הניווט של הפקד הנתונים, הנתונים המוצגים בפקד איגוד הנתונים מתעדכנים, כדי שישקפו את המידע המאוחסן ברשימה הנוכחית.

אך פקדי איגוד הנתונים אינם מוגבלים להצגת הנתונים המאוחסנים ברשימה בלבד, רובם יכולים גם לשמש לצורך שינוי הנתונים. שינוי הנתונים דורש מהמשתמש לערוך את תוכן הפקד. הנתונים המאוחסנים במסד הנתונים מעודכנים באופן אוטומטי בעקבות כל שינוי בתוכן פקד איגוד הנתונים או כתוצאה מסגירת הטופס.

הערה:

עקב העובדה שפקדי Label (תווית) אינם כוללים קטע הניתן לעריכה, המידע המוצג במסגרת התווית אינו ניתן לשינוי. גם פקדים המצויים במצב נעילה, או במצבים אחרים המונעים עריכת נתונים, לא מאפשרים למשתמשים לערוך נתונים המוצגים בהם.

כל סוג פקד איגוד נתונים משמש לעריכה ו/או הצגת נתונים מסוגים שונים. פקדי איגוד נתונים מאפשרים לך לטפל במחרוזות, מספרים, תאריכים, ערכים לוגיים ואפילו בתמונות ובתזכירים. טבלה 25.1 תפרט את חמשת הסוגים הבסיסיים של פקדי איגוד נתונים ואת סוגי הנתונים שהם משמשים לטיפול בהם. הטבלה גם תפרט את מאפיין הפקד שבו מאוחסנים הנתונים הרלוונטיים.

טבלה 25.1: הפקדים המשמשים לטיפול בנתונים מסוגים שונים

שם הפקד	סוגי נתונים	מאפיין הפקד
Label	Text, Numeric, Date	Caption
TextBox	Text, Memo, Numeric, Date	Text
CheckBox	Logical, True/False	Value
PictureBox	Long Binary	Picture
Image	Long Binary	Picture

הוספת פקדים לטפסים

להוספת פקד איגוד נתונים לטופס, בחר את הפקד בארגז הכלים ושרטט אותו על הטופס. תרשים 25.5 יציג תיבת טקסט שהוספה לטופס המכיל את פקד הנתונים. שים לב שמאפיין Caption של פקד הנתונים שונה גם הוא.



תרשים 25.5: שרטט פקדי איגוד נתונים על טופס, כפי שהיית משרטט כל פקד אחר



טיפ:

אם תחזיק את מקש Ctrl לחוץ, בזמן שתלחץ על פקד בארגז הכלים, תוכל להוסיף לטופס שלך כמה מופעים של הפקד. כך לא תידרש ללחוץ פעמים רבות על הלחצנים הכלולים בארגז הכלים. כאשר תסיים להוסיף את מופעים הדרושים לך, לחץ על כלי מצביע העכבר בארגז הכלים.

מובן מאליו שלא מספיק לשרטט את הפקד כדי לקשר אותו למסד הנתונים. תחילה תידרש להגדיר שני מאפיינים נוספים.

שימוש בפקד איגוד נתונים להצגת מידע

כדי שיתאפשר לפקד איגוד נתונים לעבוד עם הנתונים המאוחסנים במערך רשומות, תידרש לקשר את הפקד לפקד הנתונים המייצג את אותו מערך רשומות (זכור שכבר למדת כיצד מקשרים פקד נתונים לנתונים) ואל שדה מסוים במערך הרשומות. תחילה יהיה עליך להגדיר את מאפיין DataSource של פקד איגוד הנתונים. הגדרת מאפיין זה מקשרת בין פקד האיגוד (פקד TextBox - תיבת הטקסט במקרה שלנו) לבין פקד הנתונים. לאחר מכן יהיה עליך להגדיר את מאפיין DataField של פקד האיגוד, המקשר בינו לבין שדה מערך הרשומות שיוצג במסגרתו. כאשר תמשיך בעבודה על היישום לדוגמה BIBLIO, שביצירתו התחלנו קודם לכן, תלמד אודות אופני הפעולה של פקדי איגוד שונים.

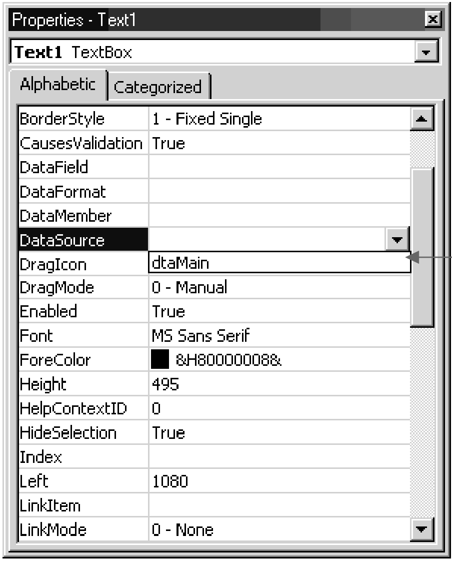
הגדרת מאפיין DataSource

כדי להגדיר את מאפיין DataSource, בחר אותו בחלון המאפיינים של הפקד. לחץ על החץ הנפתח המוצג מימין לשורת המאפיין, כדי להציג רשימת פקדי נתונים המוצגים על הטופס הפעיל. הגדר את מאפיין DataSource על ידי בחירת אחד הפקדים המוצגים ברשימה. תרשים 25.6 מתאר ביצוע פעולה זו.



טיפ:

במקום לפתוח את הרשימה, תוכל ללחוץ לחיצה כפולה על שורת המאפיין DataSource, כדי להציג את פקדי הנתונים הזמינים לשימוש.

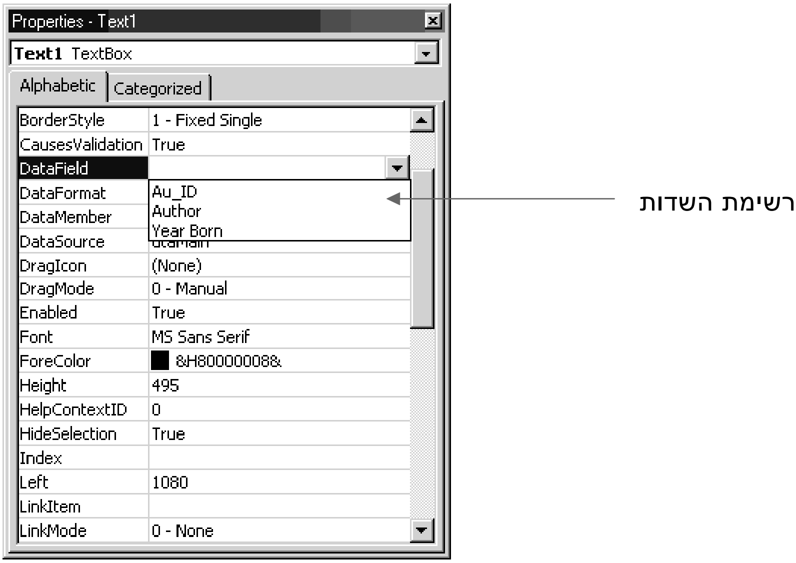


תרשים 25.6: מאפיין DataSource מגדיר את הקישור בין פקד המאפשר ביצוע פעולות על נתונים, לבין פקד הנתונים

הגדרת מאפיין DataField

מאפיין DataSource אומנם מורה לפקד האיגוד מיהו פקד הנתונים ממנו הוא אמור לקבל את הנתונים, אך עליך עדיין להגדיר לפקד האיגוד מהו הנתון שעליו לאחזר. תוכל להגדיר זאת על ידי מאפיין DataField של פקד האיגוד. מאפיין זה מורה לפקד הנתונים איזה מבין השדות שבמערך הרשומות יטופל על ידי אותו פקד איגוד.

כדי להגדיר את מאפיין DataField של הפקד, בחר מאפיין זה בחלון המאפיינים של הפקד, לחץ על החץ הנפתח המוצג מימין לשורת המאפיין ובחר את אחד הפריטים שברשימה. הרשימה תכלול את כל השדות הזמינים שבמערך הרשומות שהוגדר במסגרת מאפיין DataSource (ראה תרשים 25.7).



תרשים 25.7: הגדר את מאפיין DataField של פקד איגוד הנתונים, מתוך רשימת השדות המוגדרים במסגרת פקד הנתונים שנבחר

טיפ:

במקום להשתמש ברשימה הנפתחת, תוכל ללחוץ לחיצה כפולה על שורת המאפיין DataField, כדי להציג את השדות הזמינים לשימוש.

אזהרה:

לא יתאפשר לך לבחור הגדרה עבור מאפיין DataField, עד שתגדיר את מאפיין DataSource.

יצירת יישום פשוט

בתחילת הפרק התחלת בבנייה ניסיונית של יישום קטן, המיועד לאפשר לך להציג את שמות הסופרים שבמסד הנתונים BIBLEO ולשנותם. כעת תמשיך בבניית היישום.

הגדרת הטופס

לאחר שכבר הגדרת פקד נתונים וקישרת אותו למסד הנתונים BIBLEO תוך שימוש במאפיין DatabaseName שלו, תוכל להגדיר את מאפיין RecordSource של הפקד. בחר את הטבלה Authors מרשימת הטבלאות המוצגת בחלון המאפיינים של הפקד. בתום פעולה זו פקד הנתונים יהיה מוכן לשימוש.

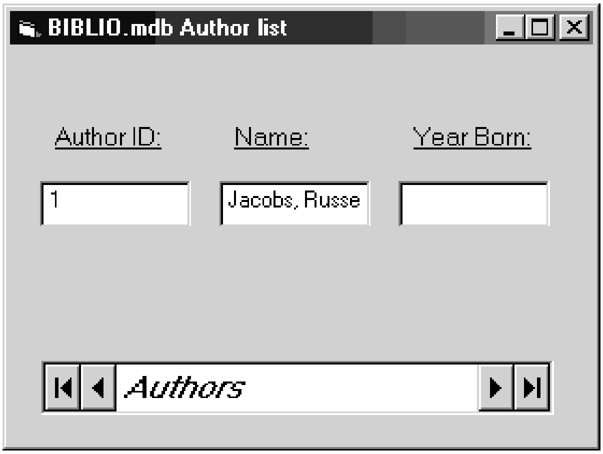
השלב הבא בתהליך יצירת טופס הגישה לנתונים הוא הוספת פקדי איגוד נתונים. כדי לפשט את הדוגמה, בחר בתיבת טקסט כדי להציג כל אחד מהשדות. הוסף לטופס פקד Label בלתי מקושר שיתאר מהו הנתון המוצג בכל אחת מתיבות הטקסט. לבניית

יישום הדוגמה יידרשו שלושה פקדי TextBox ושלושה פקדי Label. הגדר את מאפיין DataSource של כל תיבות הטקסט כ-dtaMain, שהוא שם פקד הנתונים שהוספת לטופס. יהיה עליך גם להגדיר את מאפיין DataField של כל אחת מתיבות הטקסט. זכור כי מאפיין DataField משמש לצורך קישור כל אחת מתיבות הטקסט לשדה מסוים במסד הנתונים. טבלה 25.2 מפרטת את הגדרות המאפיין DataField עבור כל אחת מתיבות הטקסט ואת הגדרת המאפיין Caption של הפקד Label התואם לכל אחת מהתיבות. בטבלה מוצגים שמות ברירת המחדל של פקדי תיבת הטקסט.

טבלה 25.2: הגדרות DataField ו-Caption של הפקדים שבטופס הגישה לנתונים

שם פקד TextBox	DataField	Caption של הפקד התואם
Text1	Au_ID	Author ID:
Text2	Author	Name:
Text3	Year Born	Year Born:

בתום הוספת פקדי האיגוד והגדרת מאפייניהם, הטופס שלך ייראה כמו זה המוצג בתרשים 25.8.



תרשים 25.8: ניתן ליצור טופס פשוט להזנת נתונים תוך שימוש בפקד הנתונים ובפקדי TextBox המוצמדים אליו

בעת שימוש בפקד נתונים רגיל, אין אפשרות להגדיר את מאפיין Data Source בזמן ריצה. אפשרות זו קיימת בעת שימוש בפקד ADO Data Control (המתואר בפרק 28, **גישה לאובייקטי נתונים באמצעות פקדי ActiveX**).

ראה: סעיף "שימוש ב-ADO Data Control", פרק 28.

ניווט במסד הנתונים

כעת, לאחר שיצרת את טופס הזנת הנתונים, תוכל לנסות אותו, על ידי הפעלת התוכנית. כאשר התוכנית תתחיל לפעול, הטופס ייטען וסביר להניח שהנתונים שברשומה הראשונה יוצגו בתיבות הטקסט. כעת תוכל לראות כיצד ניתן להיעזר בפקד הנתונים לצורך ניווט בין הרשומות המאוחסנות במסד הנתונים. תוכל לעבור אל הרשומה הראשונה במסד הנתונים, אל הרשומה האחרונה בו, אל הרשומה הקודמת או אל הרשומה הבאה פשוט על ידי לחיצה על הלחצן המתאים בפקד.

תוכנית פשוטה זו תאפשר לך גם לעדכן ולערוך את מסד הנתונים. נסה להקליד שנת לידה בשדה Year Born ולעבור אל רשומה אחרת. פעולה זו גורמת לשינוי הנתונים ברשומה הפיסית המאוחסנת במסד הנתונים המקושר אל הפקד. שינוי זה יישמר גם אם תכבה את המחשב.

רשומות אקראיות

אם תעיין במסד הנתונים תיווכח כי הרשומות הכלולות בו מסודרות בסדר אקראי לכאורה ולא בסדר האלף-בית. אך אל תיבהל, לא שגית בהגדרת הטופס, הרשומות מוצגות לפניך בהתאם לסדר הפיסי של הטבלה, כלומר לפי הסדר שבו הן הוזנו. כדי לגרום להצגת הרשומות על פי סדר האלף-בית, הצב את המחרוזת שלהלן במאפיין RecordSource של פקד הנתונים:

```
SELECT * FROM AUTHORS ORDER BY AUTHOR
```

המחרוזת שהצבת במאפיין DataSource היא דוגמה למשפט SQL.

שימוש בקוד בשילוב עם פקד הנתונים

היישומים שתיצור במציאות יהיו דינמיים יותר מיישום הדוגמה שיצרת זה עתה. לפיכך, חשוב כי תבין שעדיף להגדיר משפטי SQL ופריטי מידע לא קבועים אחרים תוך שימוש בקוד. למרבה המזל, פקד הנתונים מאפשר לך להיעזר בקוד לצורך ביצוע פעולות על רוב סוגי האובייקטים עליהם הוא מתבסס.

כדי להמחיש זאת, נמשיך בפרויקט הדוגמה. הוסף לטופס שני פקדי Command Button (לחצני פקודה) שתגדיר עבורם את השמות cmdSortAuthor ו-cmdSortYear. הוסף את קטעי הקוד שלהלן לשגרות האירוע Click של שני הפקדים:

```
Private Sub cmdSortAuthor_Click( )
        dtaMain.RecordSource = "SELECT * FROM authors ORDER BY author"
        dtaMain.Refresh
End Sub

Private Sub cmdSortYear_Click( )
        dtaMain.RecordSource = "SELECT * FROM authors ORDER BY [Year Born] DESC"
        dtaMain.Refresh
End Sub
```

הפעל את התוכנית. כעת תוכל לשנות את הסדר בו ממוינות הרשומות על ידי לחיצה על אחד משני הלחצנים. כך תוכל לראות כי הגדרת המאפיין RecordSource, במסגרת התוכנית, מאפשרת לך גמישות רבה יותר מכפי שתוכל להשיג על ידי הגדרת המאפיין בעת עיצוב היישום.

הערה:

השימוש בקוד מאפשר לך לשלוט על היבטים נוספים של פקדי Data. לדוגמה, כעת לא תידרש להגדיר קישורים בין פקדים לבין נתונים כדי שיתאפשר לך להציג את הנתונים תוך שימוש בפקדים. תוכל להיעזר בקוד כדי לגשת למאפיין RecordSet של פקד הנתוניםבאופן ישיר:

```
MsgBox "Current Author is" & dtaMain.RecordSet.Fields("Author")
```

ראה: פרק 26, "שימוש באובייקטי גישה לנתונים (DAO)" כיצד לבצע גישה למסד הנתונים תוך שימוש בקוד בלבד.

הוספה וגריעת רשומות

בוודאי נוכחת כבר כי נוח להשתמש בפקד הנתונים. אולם, חסרות לו מספר תכונות בעלות חשיבות רבה במסגרת יישומים לניהול מסדי נתונים, כגון יכולת להוסיף ולגרוע רשומות. במצב הנוכחי, יישום הדוגמה מאפשר לך לערוך רשומות קיימות אך לא להוסיף חדשות למסד הנתונים. כדי לאפשר יכולת זו יהיה עליך לערוך מספר שינויים ביישום.

הוסף לטופס יישום הדוגמה שני פקדי Command Button והגדר עבורם את השמות cmdAddRec וכן cmdDelRec. הוסף את קטעי הקוד המובאים בתוכנית 25.1 לשגרת אירוע Click של כל אחד מהלחצנים, כדי להקנות להם את היכולות המתאימות:

תוכנית 25.1: BIBLIOSAMPLE.VBP - הוספת שיפורים ליישום הדוגמה.

```
Private Sub cmdAddRec_Click( )
        dtaMain.Recordset.Addnew
End Sub
Private Sub cmdDelRec_Click( )
        dtaMain.Recordset.Delete
        If Not dtaMain.RecordSet.Eof Then
                dtaMain.RecordSet.MoveNext
        Else
                dtaMain.RecordSet.MoveLast
        End If
End Sub
```

בפרק 26 תלמד כי בעת הוספת רשומה חדשה או עריכת רשומה קיימת, Visual Basic עוברת בין מספר שלבי פעולה:

1. **Add** (הוספה). בעת הוספת רשומה חדשה, נוצרת רשומה ריקה ב**מאגר העתקה** (Copy Buffer), שהוא מקום בו נערכות רשומות לפני שהן מוספות למסד הנתונים. לאחר מכן תוכל לערוך את הרשומה החדשה.
2. **Edit Mode** (מצב עריכה). בזמן ביצוע פעולות לשינוי המידע המאוחסן ברשומה, אתה עובד במצב עריכה של Visual Basic. בעת עבודה במצב עריכה, הרשומה כבר הועתקה אל מאגר ההעתקה וייתכן שהיא כבר שונתה על ידי התוכנית, אך היא טרם עודכנה במסד הנתונים המקושר לתוכנית.
3. **Update** (עדכון). לאחר שתסיים לערוך את ערכי השדות, מאגר ההעתקה יועבר למסד הנתונים, עקב שינוי ברשומה הפעילה או עקב הפעלת השיטה Update.

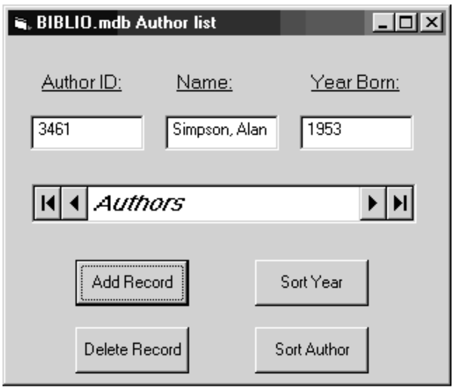
ניתן לראות כי בתוכנית הדוגמה 25.1 לא נכללת פקודה כלשהי להפעלת השיטה Update (פקדי Data מבצעים פעולות Update, באופן אוטומטי בעקבות מעבר אל רשומה אחרת או סגירת טופס).

הערה:

את השיטות MoveNext ו-MoveLast הוספת ללחצן המחיקה כדי לכפות מעבר לרשומה חדשה. לאחר שרשומה נגרעת ממסד הנתונים, לא מתאפשרת גישה אליה, אך היא מוצגת על המסך עד שמבוצעת פעולה למעבר אל רשומה אחרת. אם לא היית כופה פעולת מעבר לרשומה אחרת והיית מנסה לגשת לרשומה שנגרעה, היתה נגרמת שגיאה.

כעת ייראה הטופס שלך כמו זה המוצג בתרשים 25.9.

בנוסף להרחבת היכולות המוצעות על ידי לחצני פקד הנתונים, באפשרותך גם להחליף לחצנים אלה. הגדר את מאפיין Visible של פקד הנתונים כ-False והוסף לטופס לחצני פקודה חדשים אשר יבצעו פעולות כגון MoveNext ופעולות ניווט אחרות המבוצעות על ידי פקדי Data.



תרשים 25.9: ניתן להוסיף יכולות חדשות לטפסי הזנת נתונים על ידי הקצאת פקודות תוכנית ללחצני פקודה

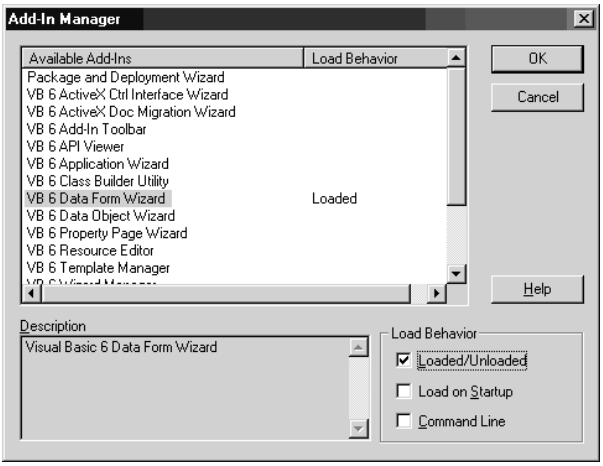
יצירת טפסים באופן אוטומטי

פקדי איגוד נתונים מאפשרים ליצור טפסים להזנת נתונים, בהשקעת מאמץ מינימלי.

למעשה אתה יכול ליצור טפסים כאלה באופן פשוט עוד יותר על ידי שימוש ב- Data Form Wizard (DFW - אשף טפסי הנתונים). DFW הוא אחד התוספות (Add-ins) המצורפות ל- Visual Basic. השימוש בו יאפשר לבחור מסד נתונים ומקור רשומות. לאחר מכן האשף ייצור טופס הזנת נתונים, באופן אוטומטי. סביר להניח שהטופס שהאשף ייצור לא יהיה בדיוק הטופס הרצוי לך, אך תוכל לשנות את עיצוב הטופס בקלות ולשמור את השינויים שביצעת. השימוש בכלי DFW הוא דרך טובה מאוד ליצירה מהירה של הטפסים הדרושים לאב-טיפוס או ליישום פשוט.

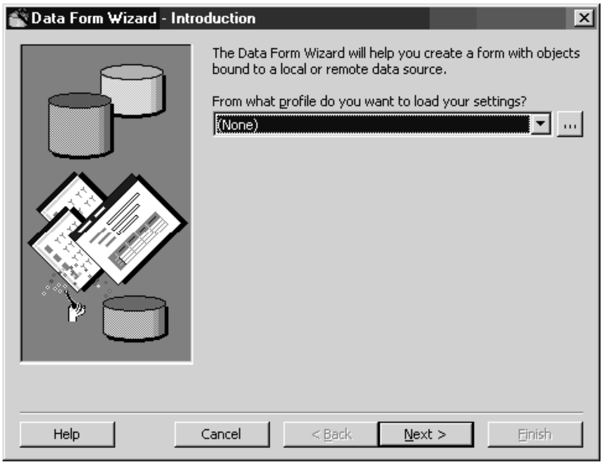
הגדרת Data Form Wizard

כדאי לחזור ולהזכיר ש-DFW הוא האחד התוספות המצורפות ל- Visual Basic. אולם, בעת הצגת תפריט Add-ins, הפריט VB 6 Data Form Wizard אינו מופיע. עליך להודיע לסביבת הפיתוח של Visual Basic כי אתה מעוניין בגישה לכלי עיצוב הטפסים (Form Designer). תוכל לעשות זאת על ידי הפעלת הפקודה Add-in Manager מתפריט Add-ins. פקודה זו גורמת להופעת תיבת הדו-שיח Add-in Manager המוצגת בתרשים 25.10.



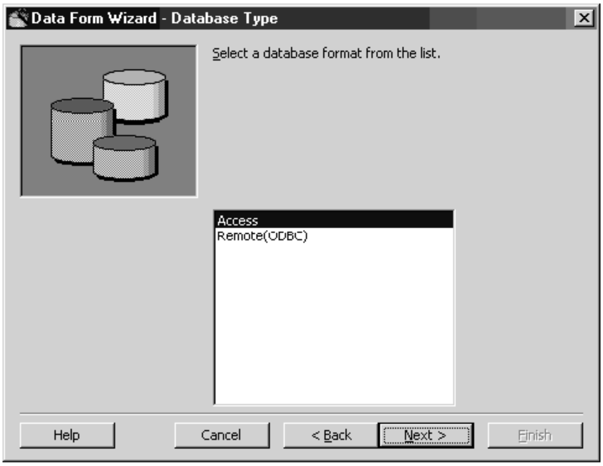
תרשים 25.10: Add-in Manager (מנהל התוספות) מאפשר לך להוסיף יכולות לסביבת העיצוב של Visual Basic

כדי לגשת ל-DFW, עליך להאיר את הפריט **VB 6 Data Form Wizard** ברשימה וללחוץ על תיבת הסימון **Loaded/Unloaded**. המילה Loaded תוצג בטור הימני של רשימת התוספות. כעת לחץ על לחצן OK ובזאת סיימת את פעולת ההוספה. כאשר תציג את תפריט Add-ins, הפריט DFW יוצג. בחירה בפריט זה תפתח את תיבת הדו-שיח Introduction של האשף המוצגת בתרשים 25.11. מסך זה מכיל מעט מידע על האשף ומאפשר לך לטעון אפשרויות שהגדרת במהלך עבודתך הקודמת עם האשף.



תרשים 25.11: Data Form Wizard **יוצר עבורך טפסי הזנת נתונים באופן אוטומטי**

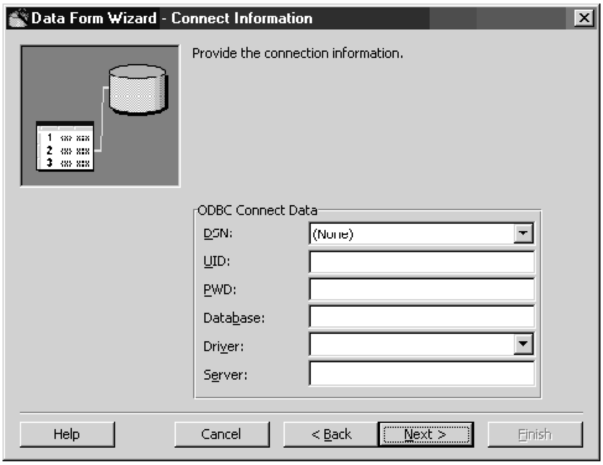
לחיצה על לחצן Next של טופס הפתיחה תעביר אותך אל מסך Database Type (סוג מסד נתונים) של האשף. מסך זה, המוצג בתרשים 25.12, מאפשר לך לבחור את סוג מסד הנתונים אליו תוכל לגשת באמצעות הטופס. לבחירת סוג, לחץ על שם הסוג הרצוי ברשימה, ולאחר מכן לחץ על לחצן Next כדי להמשיך ביצירת הטופס. אם אתה עוקב אחר הדוגמה המוצגת בפרק זה, ודא שבחרת את הפריט Access ולחץ Next.



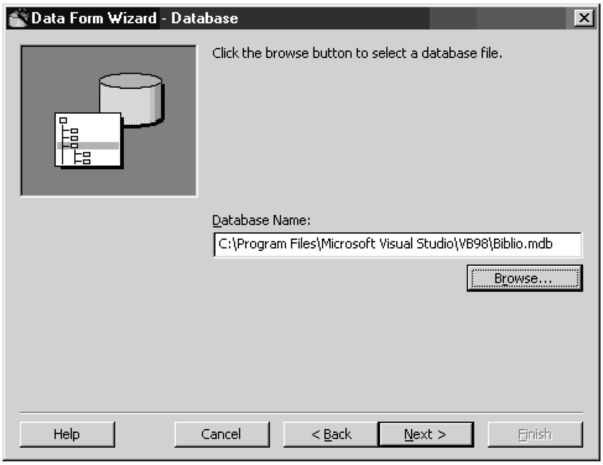
תרשים 25.12: תוכל לבחור ליצור טפסים תוך שימוש בסוגים נפוצים של מסדי נתונים

לאחר שתבחר את סוג מסד הנתונים הרצוי לך, יהיה עליך לבחור את מסד הנתונים עצמו ואת מקור הרשומות (Record Source) עמם תעבוד. את מסד הנתונים ומקור הרשומות תבחר באמצעות מסך Database של Data Form Wizard. אם תבחר באפשרות ODBC (Open Data Base Connectivity - ממשק הקישוריות הפתוחה של Microsoft), תתבקש להגדיר את כל ההגדרות הדרושות לשם התקשרות למקור נתונים המיישם את ממשק ODBC (ראה תרשים 25.13). אם תבחר באפשרות Access, אשר מאפשרת לך

להשתמש במסד הנתונים BIBLEO, תתבקש להזין שם קובץ מסד נתונים (ראה תרשים 25.14). לצורך הדוגמה שלנו יהיה עליך להיעזר בלחצן Browse כדי לאתר את קובץ מסד הנתונים BIBLEO.MDB, המאוחסן בתיקיה הראשית של Visual Basic.



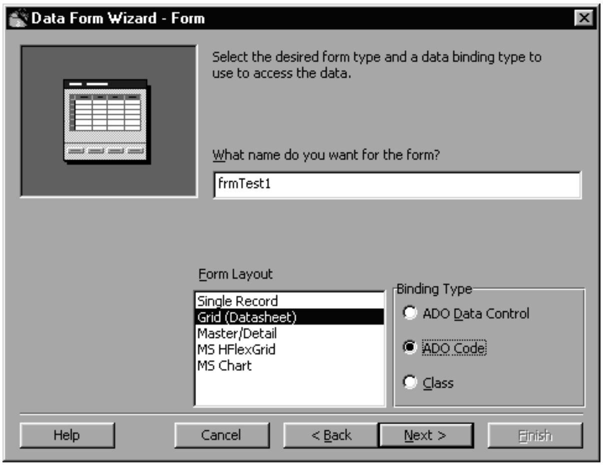
תרשים 25.13: הזן את כל הנתונים הדרושים להגדרת קישור למסד נתונים מסוג ODBC



תרשים 25.14: בעת שימוש במסד נתונים של Access, תידרש להגדיר רק את שם קובץ מסד הנתונים ואת נתיב הגישה אליו

גישה למקור הנתונים

לאחר בחירת מסד הנתונים, עבור למסך Form של האשף (תרשים 25.15). מסך זה יאפשר להגדיר שם עבור הטופס ולבחור את האופן בו יוצגו הנתונים.



תרשים 25.15: המסך Form יאפשר לך לבחור את מבנה הטופס הרצוי לך

יוצעו לך חמישה סוגי טפסים, מביניהם תוכל לבחור:

- **Single Record**. טופס כזה יאפשר לך לערוך את הנתונים שבמערך הרשומות, רשומה אחר רשומה. זה הסוג הנפוץ לטפסים להזנת נתונים.
- **Grid** (Datasheet). טופס כזה יאפשר לערוך מספר רשומות בבת-אחת. טופס כזה דומה למבט מערך רשומות ב-Access או גיליון של Excel.
- **Master/Detail**. מאפשר לערוך את הנתונים שברשומת אב יחידה עם הנתונים שברשומות הצאצא שלה. לדוגמה, בטופס כזה ניתן להשתמש לצורך הצגת נתונים של הזמנה לצד נתוני הפריטים שבה.
- **MS HflexGrid**. אפשרות זו מציגה טופס בעל מבנה רשת (Grid) תוך שימוש בפקד חדש מסוג Hierarchical FlexGrid.
- **MS Chart**. מציג תרשים המבוסס על הנתונים הרלוונטיים.

הבחירה בסוג טופס מסוים תשפיע לא רק על חזות הטופס, אלא גם על בחירת הרשומות שתוצגנה במסגרתו. בעת שימוש בטופס מסוג Single Record או Grid, תידרש לבחור מקור רשומות אחד. בעת שימוש בטופס מסוג Master/Detail תידרש לבחור שני מקורות רשומות. אך לא תידרש להתייחס לנושא זה מפני שהאשף ידריך אותך בביצוע הפעולות הדרושות. הרי לשם כך נועדו אשפים.

אזהרה:

אם תיצור טופס מסוג Master/Detail, תידרש תחילה להגדיר קשר בין הטבלאות שתבחר. מידע הקשר ישמש לשם שמירה על סינכרוניזציה של הנתונים שיוצגו.

לצורך פרויקט הדוגמה, הגדר את שם הטופס כ-frmTest1 ואת המבנה כ-Grid.

בחירת סוג האיגוד

פקד הנתונים הרגיל, אשר מצוי בשוק מזה זמן רב, נעזר באובייקטים מסוג Data Access Object (DAO). אך סוג אובייקט זה מצוי בסוף דרכו והוא יוחלף על ידי אובייקטים מסוג ActiveX Data Object (ADO). עדות לכך ניתן למצוא בעובדה שהאשף של Visual Basic 6 תומך אך ורק ביצירת טפסים הנעזרים באובייקטי ADO. תיבת ההגדרה BindingType (סוג איגוד) שבמסך Form מאפשרת ליצור טפסי הזנת נתונים משלושה סוגים:

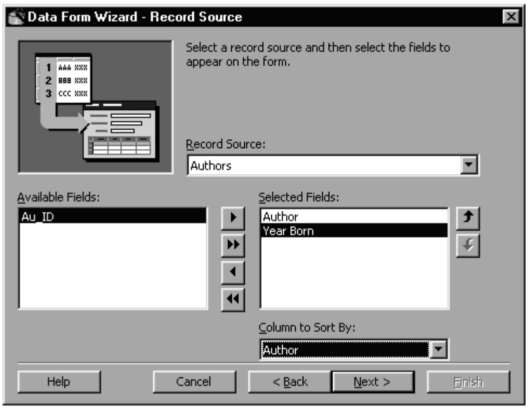
- **ADO Data Control**. הגדרה זו תיצור טופס ובו פקדים מסוג ADO Data.
- **ADO Code**. הגדרה זו תאפשר גישה למסד הנתונים רק על ידי שימוש בקוד.
- **Class**. הגדרה זאת נעזרת במודול מחלקה (Class Model) לצורך גישה לנתונים.

כל השיטות האלו נעזרות באובייקטים חדישים מסוג ActiveX Data Objects.

לצורך פרויקט הדוגמה שלנו, בחר בסוג האיגוד ADO Code.

בחירת שדות בעזרת Data Form Wizard

לאחר שתבחר את מסד הנתונים ואת סוג הטופס הרצוי, לחץ על Next כדי להציג את מסך Record Source של האשף, המוצג בתרשים 25.16. מסך זה הוא אמצעי ידידותי המסייע בבחירת הטבלה או השאילתה שעליהם יתבססו הטופס והשדות שבמסגרתו.



תרשים 25.16: בחר את מקור הרשומות ואת השדות שיוצגו בטופס

כדי להגדיר את השדות שיוצגו בטופס יהיה עליך לבצע את הפעולות הבאות:

1. בחר את מקור הרשומות (טבלה או שאילתה) מהתיבה המשולבת.
2. בחר את השדות שייכללו בטופס על ידי לחיצה על שמות השדות ברשימה **Available Fields** (שדות זמינים לבחירה). תוכל ללחוץ לחיצה כפולה על שדה כדי לבחור אותו או להאיר שדה וללחוץ על לחצן הבחירה (>).
3. סדר את השדות בסדר הרצוי לך על ידי שינוי סדר הפריטים הכלולים ברשימה **Selected Fields** (שדות שנבחרו). תוכל לשנות את סדר השדות על ידי הארת שם שדה ולחיצה על לחצן ההעלאה או ההורדה (פעולה זו היא אופציונלית).
4. בחר את העמודה שלפיה ימוין מערך הרשומות, על ידי בחירת העמודה בתיבה המשולבת **Column To Sort** (פעולה זו היא אופציונלית).
5. לחץ על הלחצן **Next** כדי לעבור אל המסך הבא.

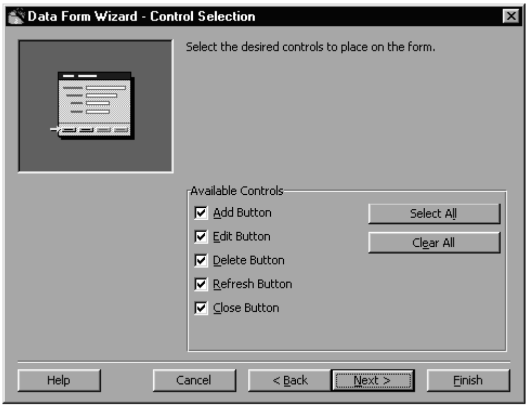
לצורך יישום הדוגמה, בחר את הטבלה **Authors** מהרשימה הנפתחת. בצע את הפעולות שתוארו כאן כדי לצרף לטופס את השדות **Author** ו-**Year Born**.

בחירת פקדים

לאחר שתבחר את כל השדות שאתה מעוניין להציג בטופס, תידרש לבצע מספר פעולות בחירה נוספות - אשר תגדרנה את חזות הטופס. תוצגנה לפניך מספר תיבות דו-שיח, בהתאם לסוג הטופס שהגדרת במסך Form. אחת מתיבות דו-שיח אלו תאפשר לך לבחור לחצנים שאתה מעוניין שיוצגו על הטופס. זוהי תיבת הדו-שיח **Control Selection** של אשף DFW המוצגת בתרשים 25.17.

טבלה 25.3 תפרט את הלחצנים שאותם תוכל להציג על טפסים להזנת נתונים. חלק מלחצנים אלה לא יהיה זמין לבחירה בעת שימוש בסוגים מסוימים של טפסים.

לאחר שתגדיר את ההגדרות בתיבות הדו-שיח הקשורות לעיצוב הטופס, לחץ על הלחצן Next ותוצע לך אפשרות לשמור את ההגדרות כ-Profile (פרופיל). את הפרופיל הזה תוכל לטעון אחר כך מבלי שתידרש לבצע את כל הפעולות הקודמות.

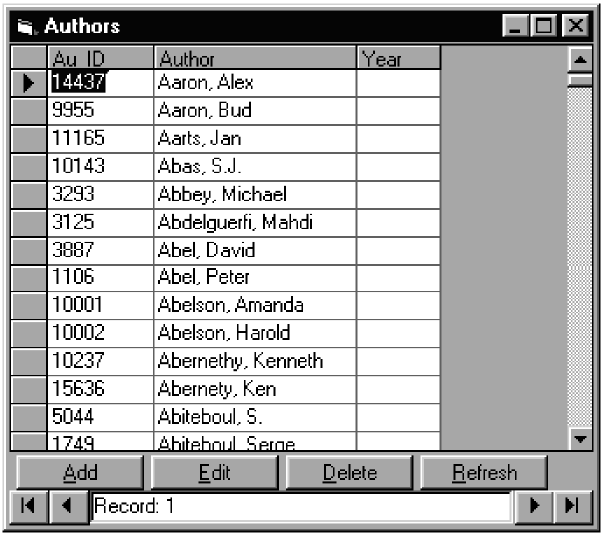


תרשים 25.17: תוכל לבחור כמה לחצני פקודה שיוצגו על הטופס

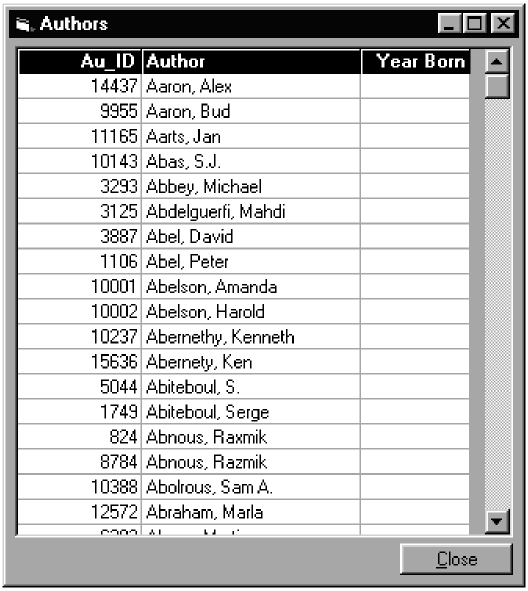
טבלה 25.3: פקדי לחצן פקודה הזמינים לשימוש והפעולות המבוצעות על-ידם

פקד זמין	פעולה
Add	מוסיף רשומה חדשה למערך הרשומות ומנקה את שדות הזנת הנתונים
Edit	מאפשר למשתמש לשנות את תוכן הרשימה המצויה כעת במצב בחירה
Update	מאחסן שינויים שבוצעו בשדות ברשומה הפעילה במסד הנתונים
Delete	גורע את הרשימה הפעילה ממסד הנתונים
Refresh	גורם לפקד הנתונים לבצע שוב את השאילתה ששימשה ליצירתו. פעולה זו דרושה רק בסביבה מרובת משתמשים
Close	סוגר את טופס הזנת הנתונים ופורק אותו מהזיכרון
Show Data Control	אם בחרת את הטופס (Datasheet) Grid ואת סוג האיגוד ADO Data Control, בחירה באפשרות זו תוסיף לטופס לחצן אשר יאפשר למשתמש להציג את פקד ADO Data Control

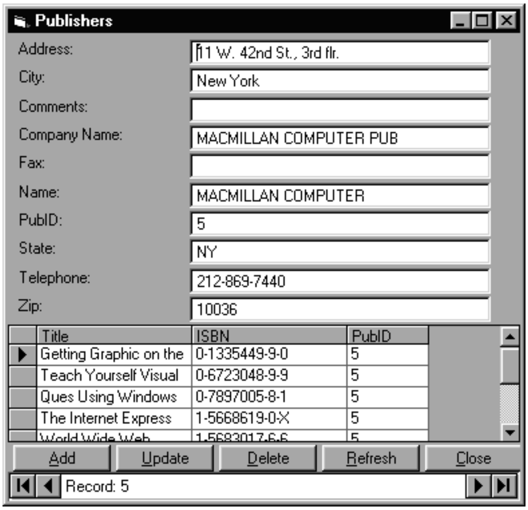
הינך עומד לפני ביצוע השלב האחרון של אשף DFW, יצירתו הממשית של הטופס. לחץ על לחצן Finish כדי להתחיל בתהליך היצירה. כעת תוכל לנוח לרגע בזמן שהאשף מבצע את פעולתו. עם סיום פעולת האשף, התוכנית שלך תכלול טופס הזנת נתונים חדש. כדי שתוכל להשתמש בטופס, כל שנותר לך הוא להשיב על מספר שאלות ולבצע פעולות בחירה אחדות. תרשימים 25.18 עד 25.20 יציגו לפניך כמה סוגי טפסים, אותם תוכל ליצור בעזרת האשף DFW.



תרשים 25.18: טופס רגיל זה, מסוג Grid נוצר בעזרת Data Form Wizard



תרשים 25.19: טופס מסוג FlexGrid אלגנטי יותר, גם נוצר בעזרת Data Form Wizard



תרשים 25.20: גם טופס זה, שהוא מסוג Master/Detail, נוצר בעזרת Data Form Wizard

מכאן...

בפרק זה למדת להשתמש בפקד הנתונים ובפקדי איגוד נתונים לצורך יצירה מהירה של יישום לטיפול במסד נתונים, הפועל על מסד נתונים קיים. תוכל למצוא מידע נוסף לגבי נושאים רלוונטיים במקומות הבאים:

- מידע אודות עיצוב מסדי נתונים ונורמליזציה של מסדי נתונים תוכל למצוא בפרק 24 **יסודות מסד הנתונים**.
- מידע אודות אופנים מתקדמים יותר לקישור תוכניות למסדי נתונים תוכל למצוא בפרק 26 **שימוש באובייקטי גישה לנתונים (DAO)**, פרק 27 - **גישה לאובייקטי נתונים מרוחקים (RDO)**, ובפרק 28 **גישה לאובייקטי נתונים באמצעות פקדי ActiveX**.
- מידע אודות יצירה של דוחות על-סמך נתונים המאוחסנים במסדי נתונים תמצא בפרק 29, **יצירת דוחות**.

26

שימוש באובייקטי גישה לנתונים (DAO)

מה בפרק?

- מבוא ל-DAO
- הגדרת פרויקט הכולל אובייקטי DAO
- פתיחת מסד נתונים קיים
- בחירת סוג מערך הרשומות שבו תשתמש
- הצבת המידע על המסך
- הגדרת מיקום מצביע הרשומות
- שימוש במסננים, אינדקסים ומיונים
- התייחסות לתוכניות המשנות רשומות מרובות
- הבנת פקודות תכנות נוספות
- מבוא לעיבוד טרנזקציות

בפרק 25 **פקד הנתונים ופקדי קישור נתונים** למדת כיצד לפתח יישום לניהול מסד נתונים במהירות, על ידי שימוש בפקד נתונים ו**בפקדי איגוד נתונים** (Data-Bound Controls) הנתמכים על ידי Visual Basic. בפרק למדת שניתן ליצור טופס הזנת נתונים כמעט מושלם פשוט על ידי הגדרת כמה מאפיינים. טפסי הזנת נתונים המבוססים על פקדי נתונים הם **כמעט** מושלמים מפני שפקדי נתונים אינם יכולים לבצע פעולות מסוימות הנדרשות לשם עבודה עם מסדי נתונים, אלא אם כן מצרפים אליהם קטעי קוד. פעולות אלו כוללות הוספת רשומות למסדי נתונים, גריעת רשומות ממסדי נתונים ואיתור רשומות מסוימות.

פעולות אלו תחשופנה אותך לנושא כתיבת קוד במסגרת יישומים לניהול מסדי נתונים. אך תוכל גם לפתח יישומים כאלה אך ורק על ידי תכנות, ללא שימוש בפקדי נתונים. כאשר תשתמש רק בפקודות תוכנית, יהיה עליך להיעזר באובייקטי גישה לנתונים - Data Access Objects (DAO).

בפרק זה תלמד כיצד ניתן להיעזר באובייקטי גישה לנתונים של Visual Basic ליצירת יישום שלם ויציב לניהול נתונים. אובייקטי גישה לנתונים משמשים כמייצגים פנימיים של **נתונים פיסיים** (Physical Data), כלומר נתונים המאוחסנים בסוגים מסויימים של מסדי נתונים או מנגנוני ניהול נתונים. ניתן להתייחס לאובייקטי הגישה לנתונים כלמשתנים מסוג מיוחד. אך "משתנים" אלה מייצגים פריטי מידע המאוחסנים *מחוץ* לתוכנית ולא בזיכרון המחשב בעת שהתוכנית פועלת.

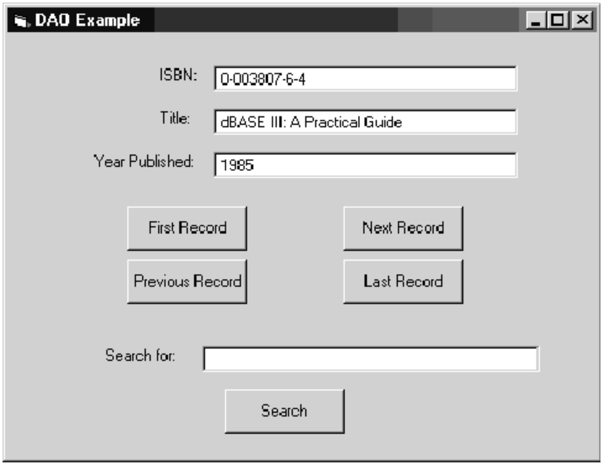
מבוא ל-DAO

שימוש באובייקטי גישה לנתונים ובפקודות התוכנית הקשורות בעבודה עמם מורכב יותר משימוש בפקדי נתונים ובפקדי איגוד נתונים, אך הוא מציע גמישות תכנותית רבה. אובייקטי גישה לנתונים והפקודות הנלוות אליהם מהווים גם את הבסיס לרוב הפעולות הקשורות בשימוש בפקדי נתונים ובפקדי קישור נתונים, כך שהם יסייעו בהבנת מושגים הקשורים בשימוש בפקדים כאלה. בפרק 25 למדת שגם בעת שימוש בפקדי נתונים, תידרש לכתוב קוד לשם הרחבת מיגוון היכולות של פקדים כאלה.

לשם המחשת נקודות הדמיון וההבדלים הקיימים בין אובייקטי גישה לנתונים לבין פקדי נתונים, תלמד בפרק זה כיצד לבנות טופס הזנת נתונים שיעבוד עם מסד נתוני הדוגמה BIBLIO.MDB (אשר מסופק יחד עם Visual Basic). טופס הזנת הנתונים שתבנה במהלך לימוד הפרק מתואר בתרשים 26.1.

אחת הסיבות העיקריות שבעטיין כדאי להשתמש בפקודות תוכנית היא שפקודות אלו מקנות לך רמה גבוהה יותר של גמישות מכפי שיתאפשר לך להשיג על ידי שימוש בפקד הנתונים. תוכל לבצע פעולות לווידוא תקפות הנתונים (Data Validation) שתהיינה מורכבות יותר מפעולות דומות שתבוצענה תוך הסתמכות על כללים המוגדרים במסגרת מסד הנתונים, מפני שפקודות תוכנית לא ניגשות למסד הנתונים באופן ישיר. תוכל גם לבטל פעולות עריכת נתונים מבלי שתידרש לבצע **טרנזקציות** (Transactions). במילים אחרות ניתן לומר שהתוכנית שלך תוכל לבדוק את הנתונים שיוזנו **לפני** שיתבצע ניסיון להוספת הנתונים למסד הנתונים. פקודות תוכנית מהוות גם אמצעי

יעיל לביצוע פעולות קלט וחיפוש, שאינן דורשות אינטראקציה עם המשתמש. דוגמאות לפעולות אלו הן קבלת נתונים ממכשירי מדידה אוטומטיים, באמצעות מודם או חיפוש מחירי פריטים בטבלה. פקודות תוכנית מאפשרות לך גם לבצע פעולות **עיבוד טרנזאקציות** (Transaction Processing).



תרשים 26.1: תוכל לבנות את טופס הזנת הנתונים המוצג כאן על ידי ביצוע הוראות שתובאנה במהלך הפרק

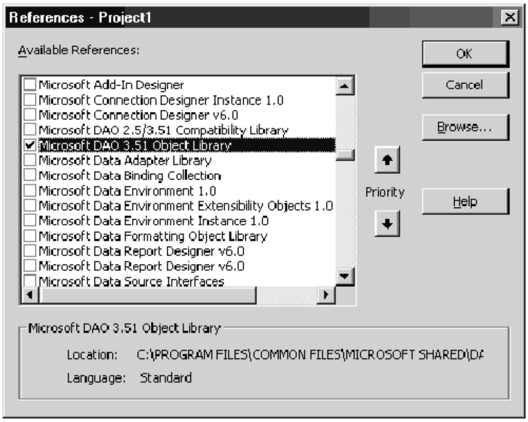
הגדרת פרויקט הכולל אובייקטי DAO

בטרם יתאפשר להוסיף לפרויקט יכולות כלשהן לניהול מסדי נתונים (פרט ליכולות הנתמכות על ידי פקדי נתונים ופקדי איגוד נתונים), תידרש לקשר את התוכנית לאחד מסוגי הספריות Data Access Objects.

להגדרת ההפניה (Reference) לספריה יהיה עליך להפעיל את הפקודה **References** מהתפריט **Project** ולבחור מתיבת הדו-שיח **References** (שבכותרת שלה יוצג גם שם הפרויקט) את אחד הסוגים Microsoft DAO Library (ראה תרשים 26.2).

הערה:

Visual Basic יכולה לכלול שני סוגי ספריות Jet DAO ביישומים. סוגי הספריות החיצוניות הם Microsoft DAO 3.51 ו- Microsoft DAO 2.5/3.51. אם אתה מתכנן שכל יישומי הלקוח שלך יהיו בני 32 סיביות ושתיעזר רק בגרסת 32 סיביות, מנגנון Jet (אשר נתמך רק על ידי Access 95/97), בחר בספריית DAO 3.51. אם תידרש לבצע חילופי מידע עם מערכות בנות 16 סיביות או עם יישומים המבוססים על Access 2, יהיה עליך להשתמש בספריית התאימות DAO 2.5/3.51.



תרשים 26.2: הוספת הפניה לספריית DAO הופכת את אובייקטי DAO זמינים לתוכנית

בעת פעולת תוכנית, מסד נתונים נפתח כחלק מ**הפעלה** (Session) עם **מנגנון מסד נתונים** (Database Engine). ב- Visual Basic מנגנון מסד הנתונים מיוצג על ידי אובייקט מסוג DBEngine. הפעלה מוגדרת על ידי יצירת אובייקטי Workspace המוגדרים במסגרת אובייקט DBEngine. במסגרת אובייקט DBEngine ניתן להגדיר אובייקט Workspace אחד, או יותר. כברירת מחדל, הוספת הפניה לספריית DAO יוצרת במסגרתו באופן אוטומטי אובייקט DBEngine ואובייקט Workspace אשר זוכה לכינוי Workspaces(0) (השם הזה נובע מכך שזהו האובייקט הראשון הכלול באוסף האובייקטים Workspaces - שבמסגרתו מיושמת מניית אובייקטים מ-0). אובייקטי ברירת המחדל האלה יספיקו לך לצורך ביצוע רוב הפרויקטים המבוססים על אובייקטי DAO.

לאחר שתגדיר הפניה לספריית DAO תוכל לפתוח את מסד הנתונים על ידי שימוש בשיטה OpenDatabase אובייקט Workspace. סדרת המושגים שהובאה כאן עלולה לגרום לבלבול מסוים, מפני שמשתמע ממנה שיצירת אובייקט Database מתבצעת על ידי פתיחת Database קיים. השיטה OpenDatabase שייכת לאובייקט מסוג Workspace, אך בעת פנייה אליה מניחים שנעשה שימוש באובייקט Workspace המהווה ברירת מחדל - DBEngine.Workspaces(0) ולכן ניתן להשמיט את שם האובייקט המשמש לשם פנייה לשיטה. כך שבהנחה שהפרויקט מכיל רק אובייקט אחד מסוג Workspace, שתי שורות הקוד הבאות הן זהות:

```
Set dbTest = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")
Set dbTest = OpenDatabase("BIBLIO.MDB")
```

במהלך פרק זה נבנה יישום לדוגמה שיתבסס על מסד הנתונים BIBLIO.MDB, המסופק עם Visual Basic. התחל את יצירת היישום על ידי ביצוע הפעולות הבאות:

1. הפעל את Visual Basic וצור פרויקט חדש מסוג **Standard EXE**.
2. שנה את שם הטופס Form1 לשם **frmDAOTest** ושמור את הפרויקט.

3. הוסף לפרויקט הפניה לספריית DAO המתאימה, על ידי בחירת הפקודה **Refrences** מהתפריט **Project**, תוצג לפניך תיבת הדו-שיח **Refrences ProjectName** (הביטוי ProjectName יוחלף על ידי השם שתגדיר עבור הפרויקט שלך, ראה תרשים 26.2).
4. סמן ליד הפריט **Microsoft DAO 3.51 Object Library**.
5. לחץ על **OK** כדי לסגור את תיבת הדו-שיח.

לאחר שתשלים ביצוע פעולות אלו הפרויקט שלך יכיל הפניה לספריית Data Access Objects, אשר תאפשר לך להשתמש באובייקטי DAO במסגרת הפרויקט.

פתיחת מסד נתונים קיים

הצעד הראשון בכתיבת יישומים רבים המנהלים מסדי נתונים הוא התקשרות למסד הנתונים שבו מעוניינים להשתמש. אם מסד היישום שלך יפנה למסד נתונים קיים, יהיה עליך להגדיר במסגרת התוכנית אובייקט מסוג Database, ולהיעזר בו לצורך הגדרת קישור למסד נתונים קיים. בכך, אתה למעשה "פותח את מסד הנתונים" לצורך שימוש התוכנית שלך. רוב הסוגים האחרים של אובייקטי גישה לנתונים "נובעים" מתוך אובייקטים מסוג Database.

תוכל להתייחס לאובייקט מסוג Database כאל האופן בו מיוצג מסד נתונים פיסי כלשהו במסגרת התוכנית שלך ולא כלמסד הנתונים עצמו. אובייקט Database הכלול בתוכנית שלך הוא למעשה חלון למסד הנתונים.

המשך את בניית יישום הדוגמה, על ידי ביצוע הפעולות המפורטות להלן, שבמסגרתן תשתמש בשיטה OpenDatabase לשם יצירת אובייקט מסוג Database וקישורו למסד הנתונים BIBLIO.MDB :

1. בקטע General Declerations (הצהרות כלליות) של הטופס, הוסף את שורת הקוד הבאה, שתיצור משתנה ברמת טופס מסוג Database Object :

```
Dim dbTest As Database
```

2. הוסף את שורות הקוד הבאות לשגרת האירוע Load של הטופס. קטע הקוד הזה, אשר יתבצע בעת טעינת הטופס, ייצור אובייקט Database בשם dbTest על ידי פתיחת מסד הנתונים BIBLIO.MDB :

```
Dim sDBLocation As String
sDBLocation = "C:\Program Files\Microsoft Visual Studio\VB98\biblio.mdb"
Set dbTest = OpenDatabase(sDBLocation)
```

פקודות אלו תפתחנה מסד נתונים מסוג Jet עם ברירות המחדל read/write data access ו- shared Access. תחביר השיטה OpenDatabase מאפשר לך להגדיר שהגישה למסד הנתונים תותר רק למשתמש אחד ברגע נתון או רק לצורך קריאה (מצב שלא יאפשר לבצע פעולות עדכון). בעת התקשרות למסד נתונים שאינו מסד נתונים של Access, יתאפשר לך להגדיר את סוג מסד הנתונים.

באופן העבודה Read-Only (לקריאה בלבד) תשתמש כאשר תרצה להתיר רק חיפוש רשומות (תוכל לכלול ביישומים שלך טבלאות כגון טבלאות מיקוד, שתהיה מעוניין שמשתמשים יוכלו לעיין בהן, אך לא יוכלו לבצע בהן שינויים). כדי לפתוח מסד נתונים לקריאה בלבד, יהיה עליך לשנות את פקודת Set באופן המתואר להלן. הפרמטר הראשון לאחר שם מסד הנתונים יגדיר האם הגישה למסד הנתונים תותר רק למשתמש יחיד ברגע נתון והשני מציין האם הגישה היא לקריאה בלבד:

```
Set dbTest = OpenDatabase("C:\ZIPCODE.MDB", False, True)
```

פתיחת מסד הנתונים רק מקשרת את התוכנית שלך למסד הנתונים עצמו אך אינה מקנה למשתמש גישה לנתונים עצמם. אם תרצה לאפשר למשתמש גישה לנתונים, תידרש לפתוח אובייקט מסוג **מערך רשומות** (Recordset), שיקושר לנתונים המאוחסנים בטבלת מסד הנתונים או בכמה טבלאות בו.

בחירת סוג מערך הרשומות בו תשתמש

כאשר אתה יוצר אובייקט מערך רשומות, לפתיחת מערך רשומות במסגרת התוכנית, תוכל להגדירו כך שישמש לגישה לטבלה שלמה, לשדות מסוימים בטבלה, לרשומות מסוימות בטבלה, או לשילוב כלשהו של רשומות ושדות הלקוחים מכמה טבלאות. Visual Basic תומכת בשלושה סוגים של מערכי רשומות:

סוג המערך	סוג הנתונים שבמערך הרשומות
Table	כל הרשומות בטבלה פיסית שבמסד הנתונים
Dynaset	סדרת מצביעים המאפשרים גישה לשדות ולרשומות שבטבלה או בכמה טבלאות שבמסד הנתונים
Snapshot	עותקים לקריאה בלבד של הנתונים שבטבלה או בכמה טבלאות. הם מאוחסנים בזיכרון

הערה:

במהלך פרק זה נתייחס למושגים Table, Dynaset ו-Snapshot, אך חשוב שתזכור שכל המושגים האלה הם סוגים של מערכי רשומות, שניתן לגשת אליהם רק על ידי שימוש באובייקטים מסוג Recordset. כך שכל Dynaset הוא למעשה מערך רשומות מסוג Dynaset (Dynaset Recordset), כל Table הוא מערך רשומות מסוג Table (Table Recordset) וכל Snapshot הוא מערך רשומות מסוג Snapshot (Snapshot Recordset). גרסאות קודמות של Visual Basic תמכו באובייקטים מהסוגים Table, Dyanset ו-Snapshot אשר אינם נתמכים עוד.

הסעיפים הבאים יתארו את הסוגים השונים של מערכי רשומות תוך התייחסות ליתרונות ולחסרונות של כל סוג והדגמת פקודות המשמשות לשם גישה למערכי רשומות.

שימוש ב-Table

Table (מערך רשומות מסוג Table) הוא קישור ישיר לאחת הטבלאות שבמסד נתונים. עקב העובדה שכל הנתונים מאוחסנים בטבלאות, השימוש במערך רשומות מסוג Table מהווה את אופן הקישור הישיר ביותר לנתונים. מערכי רשומות מסוג Table הם גם היחידים התומכים באינדקסים, כך שפעולות חיפוש אחר רשומה מסוימת במערך רשומות מסוג Table תתבצענה מהר יותר מפעולות חיפוש במערכי רשומות מסוג Dynaset או Snapshot.

בעת שימוש במערכי רשומות מסוג Table, פעולות פנייה לנתונים ושינוי נתונים מתייחסות לטבלה אחת ברגע נתון, תוך התייחסות לרשומה אחת ברגע נתון. אופן פעולה זה מאפשר רמת שליטה גבוהה על הפעולות שתבוצענה על הנתונים. אך הוא אינו מאפשר לבצע פעולות המשנות רשומות בכמה טבלאות, בבת-אחת, כפי שניתן לעשות באמצעות **שאילתות פעולה** (Action Queries).

יתרונות השימוש במערכי רשומות מסוג Table

השימוש במערכי רשומות מסוג Table מאפשר להשיג מספר יתרונות:

- תוכל להיעזר וליצור אינדקסים כדי לשנות את סדר הצגת הרשומות במהלך פעולת התוכנית.
- תוכל לבצע פעולות לחיפוש רשומות מסוימות, במהירות, על ידי שימוש באינדקס המתאים ובפקודה Seek.
- שינויים בטבלה שיבוצעו על ידי משתמשים העובדים במקביל אליך או על ידי יישומים שונים, יבואו לידי ביטוי מייד ולא תידרש לבצע פעולות לרענון הטבלה, כדי שייראו.

חסרונות השימוש במערכי רשומות מסוג Table

מובן שהשימוש במערכי רשומות מסוג Table מלווה גם בחסרונות מסוימים:

- לא ניתן להגדיר מסננים (Filters) שיגבילו השפעה של פעולות מסוימות רק לרשומות העונות על קריטריונים מסוימים.
- לא ניתן להיעזר בפקודה Find בעת שימוש במערכי רשומות מסוג Table. הפקודה seek תאתר רק את הרשומה הראשונה העונה על קריטריון מסוים כך שבעת הצורך בעיבוד סדרת רשומות, אתה כמפתח תידרש לכתוב קוד שיבצע את הפעולות הרלוונטיות על הרשומות הנוספות.

ברוב המקרים תוכל להתגבר על מגבלות אלו בעזרת תכנות, אך רוב הפתרונות שאותם תידרש ליישם לא יהיו אלגנטיים. בעת הדיון בפעולות למעבר על פני מערכי רשומות ולאיתור רשומות מסוימות במערכי רשומות נתייחס לפתרונות אפשריים לבעיות אלו. נושאים אלה יידונו בהמשך הפרק.

ראה, "הגדרת מיקומו מצביע הרשומות" בהמשך פרק זה.

פתיחת Table לשימוש

לפתיחת מערך רשומות מסוג Table, יהיה עליך להגדיר אובייקט RecordSet ולפנות לשיטה OpenRecordset של אובייקט Database. ההגדרה שהמערך שייפתח יהיה מסוג Table תתבצע על ידי שימוש בקבוע DBOpenTable בפרמטרים של השיטה.

בצע את הפעולות המפורטות להלן, כדי ליצור מערך רשומות מסוג Table בשם rsTitles במסגרת היישום לדוגמה:

1. הוסף את ההצהרה **Dim rstTitles as RecordSet** לקטע **General Declarations** של הטופס.
2. הוסף את שורת הקוד הבאה לשגרת האירוע **Load** של הטופס:

```
Set rstTitles = dbTest.OpenRecordset("titles", dbOpenTable)
```

פקודה זו תפתח טבלה חדשה במסד נתוני Jet, תוך שימוש באפשרויות ברירת המחדל שהן גישה משותפת וגישה לקריאה/כתיבה. תוכל לכלול במסגרת הפנייה לשיטה OpenRecordset פרמטרים אופציונליים, כדי לפתוח את הטבלה באופן שיאפשר גישה אליה רק למשתמש יחיד ברגע נתון, או באופן שיאפשר גישה אליה לקריאה בלבד. הפרמטרים האופציונליים האלה יפורטו בטבלה 26.1.

הערה:

בכל מקרה בו כוללים שם טבלה קיימת במסגרת השיטה OpenRecordset, המנגנון יוצר מערך רשומות מסוג Table, כברירת מחדל.

טבלה 26.1: אפשרויות המשנות את אופן הגישה לטבלה

אפשרות	פעולה מבוצעת
dbDenyWrite	מונעת ממשתמשים אחרים לכתוב לטבלה, בעת שהיא בשימושך
dbDenyRead	מונעת ממשתמשים אחרים לקרוא מטבלה כשהיא בשימושך
dbReadOnly	מונעת ממך לבצע שינויים בטבלה

שימוש ב-Dynasets

Dynaset (מערך רשומות דינמי) הוא אוסף נתונים בטבלה או בכמה טבלאות. נתונים אלה מורכבים משדות נבחרים מאותן טבלאות, אשר ברוב המקרים מוצגים על-פי סדר מסוים ולאחר שסוננו בהתאם לתנאים מסוימים. מערך מסוג Dynaset פונה לרשומות שהיו קיימות בטבלה בעת יצירתו. מערכים מסוג Dynaset ניתנים לעדכון, כך שכל השינויים שמשתמשים יבצעו עליהם, יאוחסנו ויעודכנו. אך הפעולות לא תשתקפנה לאחר שמערך הרשומות כבר נוצר. עקב כך, השימוש במערכי רשומות מסוג Dynaset במסגרת סוגים מסוימים של סביבות מרובות משתמשים אינו כה יעיל.

יתרונות השימוש במערכי רשומות מסוג Dynaset

להלן יפורטו כמה מהיתרונות המוצעים על ידי מערכי רשומות מסוג Dynaset:

- מערכי רשומות מסוג Dynaset מאפשרים לשלב בין נתונים מטבלאות שונות.
- מערכי רשומות כאלה מאפשרים לך להיעזר בשיטה Find לשם איתור ועיבוד כל הרשומות העונות על קריטריונים מסוימים.
- מערכי רשומות מסוג Dynaset מאפשרים לך להגביל את מספר הרשומות, או את מספר השדות שיאוחזרו ממסד הנתונים לתוך מערך הרשומות.
- מערכי רשומות מסוג Dynaset מאפשרים להיעזר במסננים ובמאפיינים להגדרת סדר מיון לשם שינוי אופן תצוגת הנתונים.

מגבלות מערכי רשומות מסוג Dynaset

למערכי רשומות מסוג Dynaset יש גם מגבלות:

- לא ניתן להשתמש באינדקסים בשילוב עם Dynasets, כך שלא ניתן לשנות את הסדר בו תוצגנה הרשומות שבמערך רשומות מסוג Dynaset על ידי שינוי אינדקס או על ידי יצירת אינדקס חדש.
- מערכי רשומות מסוג Dynaset לא יישקפו באופן אוטומטי פעולות להוספה או גריעת רשומות שתבוצענה על ידי משתמשים או יישומים אחרים. כדי להציג שינויים כאלה, תידרש לרענן את מערך הרשומות או ליצור אותו מחדש.

הגדרת מערך רשומות מסוג Dynaset

כדי להגדיר מערך רשומות מסוג Dynaset, יהיה עליך להגדיר אובייקט Recordset באמצעות משפט Dim וליצור את מערך הרשומות על ידי שימוש בשיטה OpenRecordset ובקבוע dbOpenDynaset. בעת יצירת Dynaset, משפט SQL מהווה חלק חשוב מהפנייה לשיטה. משפט זה מגדיר את הרשומות שבמערך, את תנאי הסינון ואת תנאי הצירוף המשמשים לשם קישור בין נתונים שבטבלאות שונות.

קטע הקוד הבא ידגים כיצד ליצור Dynaset על ידי שימוש במשפט SQL שיבחר רק רשומות מסוימות וימיין אותן בסדר מוגדר. הפקודות שבדוגמה תאפשרנה גישה לאותם נתונים שאליהם ביצעת גישה בדוגמה הקודמת, אך ההבדל בין הדוגמאות הוא בסוג מערך הרשומות שנוצר במסגרתן.

```
Dim MyDB As Database
Dim MyRS As Recordset
Dim sSQL As String
Set MyDB = OpenDatabase("biblio.mdb")
sSQL = "SELECT * FROM titles WHERE title <= 'B' ORDER BY title"
Set MyRS = MyDB.OpenRecordSet(sSQL, dbOpenDynaset)
```

אם תרצה לכלול בתוך Dynaset את כל הרשומות בטבלה מסוימת אך מבלי למיין אותן בסדר מסוים, תוכל להשמיט את משפט SQL ולהשתמש רק בשם הטבלה:

```
Set rsTitles = dbTest.OpenDatabase("titles")
```

הערה:

כדאי שתכלול משפט SQL, אם ייתכן שבעתיד תרצה לשנות את הקריטריונים שישמשו לבחירת רשומות.

בעת יצירת מערך רשומות מסוג Dynaset תוכל להיעזר בכל משפט SQL חוקי אשר כולל ביצוע פעולה לבחירת רשומות. תוכל גם להגדיר אפשרויות לאופן פעולת המערך. אפשרויות אלו תפורטנה בטבלה 26.2.

טבלה 26.2: אפשרויות המשנות את אופן הגישה למערך רשומות מסוג Dynaset

אפשרות	פעולה מבוצעת
dbDenyWrite	מונעת ממשתמשים אחרים לכתוב למערך כשהוא בשימוש
dbReadOnly	מונעת ממך לבצע שינויים בטבלה
dbAppendOnly	מאפשרת להוסיף רשומות חדשות לטבלה אך מונעת ממך לעיין ברשומות קיימות או לשנות אותן
dbSQLPassThrough	מעבירה את משפט SQL שישמש ליצירת מערך הרשומות ל- ODBC Server

לדוגמה, קטע הקוד הבא יראה לך כיצד ליצור מערך רשומות מסוג Dynaset שיאפשר למשתמשים לקרוא רשומות בלבד:

```
Set rsTitles = dbTest.OpenRecordset("SELECT * FROM titles",dbOpenDynaset, _
        dbReadOnly)
```

ראה, "שימוש במשפטי SELECT", נספח 3.

הערה:

ODBC Server הוא מנגנון מסד נתונים, כדוגמת SQL Server או Oracle, אשר תואם לתקן Open Database Connectivity (ODBC). שרת כזה אמור לעבד את השאילתות אצלו ולהחזיר את תוצאות העיבוד ליישום הלקוח. **מנהלי התקן** (Drivers) ממשק ODBC אשר נכתבים לרוב על ידי יצרני מנגנון מסד הנתונים מטפלים ביצירת הקישור בין Visual Basic לבין שרת מסד הנתונים. יתרונו של ODBC הוא בכך שהוא מאפשר להתקשר לנתונים המאוחסנים בשרתי מסד נתונים מבלי להידרש להכיר את אופני הפעולה הפנימיים של אותם שרתים.

ניתן גם ליצור Dynaset מתוך Dynaset אחר. אחת הסיבות שבעטיה תרצה לבצע פעולה כזו היא כדי שתוכל להשתמש במאפיינים Filter ו-Sort של מערך הרשומות הראשון לצורך הגדרת מסנן וסדר מיון הרשומות שתיכללנה במערך הרשומות השני. על ידי יצירת אובייקט Dynaset שני אתה יוצר תת-קבוצה של הרשומות שבמערך הרשומות הראשון. ברוב המקרים, מערך הרשומות השני יהיה קטן בהרבה מהראשון, מצב אשר יאפשר לעבד את הרשומות בו מהר יותר. קטע הקוד הבא ייצור מערך רשומות מסוג Dynaset מטבלת Customers, לצורך יצירת רשימת לקוחות המתגוררים בכל רחבי ארצות הברית. ממערך זה ניצור מערך רשומות שני שיכיל רק לקוחות המתגוררים במדינת טנסי אשר ימוין על-פי מיקוד, לצורך עיבוד עתידי:

```
Dim dbMarket As Database
Dim rsCustomers As Recordset
Dim rsTNCust as Recordset
Set dbMarket = OpendataBase(sDBLocation)
Set rsCustomers = dbMarket.OpenRecordset("customers", dbOpenDynaset)
rsCustomers.Filter = "state = 'TN'"
rsCustomers.Sort = "ZIP"
Set rsTNCust = rsCustomers.OpenRecordset(dbOpenDynaset)
```

ייתכן שתתהה מדוע לא תוכל ליצור את מערך הרשומות השני, שהרשומות המאוחסנות בו הן הדרושות לך, ישירות מהטבלאות המקוריות. התשובה היא שתוכל לעשות זאת במצבים בהם תידרשנה ליישום שלך רק הרשומות המאוחסנות בטבלה השנייה. אך לשם הדגמה נתייחס למערכת לניהול רשימת חברים בגוף מסוים, שבמסגרתה תרצה שתהיה לך גישה לרשימת החברים המלאה (אשר תיווצר על ידי יצירת מערך Dynaset ראשון). אך תרצה גם שהמערכת תאפשר לך ליצור רשימת כתובות למשלוח דואר לחברים המתגוררים באזור מסוים (רשימה אשר תיווצר על ידי יצירת מערך Dynaset שני). העובדה כי המערך הראשון מכיל מצביעים לרשומות שבמערך השני תאיץ את יצירתו באופן ניכר, בהשוואה למצב יצירת Dynaset מאפס.

שימוש ב-Snapshot

משם הסוג **Snapshot** (תמונת הבזק), משתמע שמערך רשומות כזה הוא צילום מסד הנתונים בנקודת זמן נתונה. מערך רשומות מסוג Snapshot דומה למערך מסוג Dynaset בכך שהוא נוצר מטבלת בסיס באמצעות שימוש במשפט SQL, מאובייקט QueryDef, ממערך רשומות מסוג Dynaset או ממערך רשומות אחר מסוג Snapshot. מערך Snapshot שונה ממערך Dynaset בכך שהוא אינו ניתן לעדכון. שורת הקוד שתוצג להלן תדגים יצירת מערך רשומות מסוג זה:

```
Set MyRS = MyDB.OpenRecordSet(sSQL, dbOpensnapshot)
```

כעיקרון, כדאי להשתמש במערכי Snapshot לעיבוד מידע שאינו רגיש לזמן, כלומר לעיבוד מידע שבעת התייחסות אליו אין משמעות לשינויים שבוצעו ברשומות לאחר יצירת המערך. אופן השימוש העיקרי במערכי Snapshot הוא לשם יצירת דוחות ומסכי הצגת נתונים שהמידע המוצג במסגרתם הוא סטטי.

יתרונות השימוש במערכי רשומות מסוג Snapshot

מערכי רשומות מסוג Snapshot מקנים לך את היתרונות הבאים:

- למזג נתונים הלקוחים מטבלאות שונות.
- שימוש בפקודות Find לאיתור רשומות.
- יצירת המערך והניווט בו מהירים יותר מאשר בעת שימוש ב-Dynaset, מפני ש-Snapshot מורכב מעותקים של הנתונים במקום ממצביעים לנתונים.

חסרונות השימוש במערכי רשומות מסוג Snapshot

החיסרון העיקרי בשימוש במערך רשומות מסוג Snapshot הוא שהמערך אינו ניתן לעדכון. חיסרון נוסף, בעת השימוש בו לא ניתן להיעזר באינדקסים לשינוי סדר הצגת הרשומות או לשם איתור רשומות מסוימות.



אזהרה:

כדאי שתקפיד שמערכי רשומות מסוג Snapshot **יכילו** מספר רשומות קטן, זאת כדי למנוע בעיות זיכרון פנוי.

הגדרת מערך רשומות מסוג Snapshot

ניתן ליצור מערך רשומות מסוג Snapshot על ידי הגדרת אובייקט Recordset תוך שימוש במשפט Dim וקריאה לשיטה OpenRecordset תוך שימוש בקבוע dbOpenSnapShot, לשם הגדרת הרשומות שתיכללנה במערך הרשומות. גם בעת יצירת מערך זה תוכל לכלול בקריאה לשיטה פרמטרים אופציונליים. פרמטרים אלה יפורטו בטבלה 26.3.

טבלה 26.3: אפשרויות המשנות את אופן הגישה למערך רשומות מסוג Snapshot

אפשרות	פעולה מבוצעת
dbDenyWrite	מונעת מאחרים לכתוב ל-Snapshot, כשהוא בשימושך
dbForwardOnly	מאפשרת לגלול קדימה בלבד, בעת הצגת Snapshot
dbSQLPassThrough	מעבירה את משפט SQL שישמש ליצירת מערך הרשומות ל- ODBC Server לשם עיבוד

שימוש במערך רשומות מסוג Forward-only

מערך רשומות מסוג Forward-only הוא סוג מיוחד של מערך רשומות מסוג Snapshot, אשר מאפשר גלילה קדימה בלבד בעת עיון ברשומות הכלולות בו. כלומר שלא ניתן להשתמש בשיטות MoveFirst, MovePrevious ו-Find בעת עבודה עם מערכי רשומות מסוג זה. יתרונו הוא שהוא מהיר יותר ממערך רשומות מסוג Snapshot, אך במערכי רשומות מסוג זה ניתן להשתמש רק במצבים שבהם יידרש מעבר אחד בלבד על מערך הרשומות, למשל במסגרת יצירת דוחות.

כדי להגדיר מערך Forward-only יהיה עליך להשתמש בשיטה OpenRecordset שתכלול בקריאה אליה את הקבוע dbOpenForwardOnly, כמתואר בשורה הבאה:

```
Set MyRs = MyDB.OpenRecordSet(sSQL, dbOpenForwardOnly)
```

הצבת מידע על המסך

נניח שיצרת טופס להזנת נתונים תוך שימוש בפקד נתונים ובפקדי איגוד. אם תרצה להציג נתונים על גבי הטופס, תידרש בסך הכל לשרטט פקדי איגוד נתונים ולהגדיר שדות שהנתונים הכלולים בהם יוצגו במסגרת הפקדים האלה. התצוגה תתבצע באופן אוטומטי. אם תשתמש באובייקטי גישה לנתונים, התהליך יהיה מורכב יותר, במעט. תידרש עדיין להשתמש בפקדים (כגון TextBox, CheckBox וכדומה) לשם הצגת המידע, אך תידרש להגדיר את השדות שמהם יילקחו הנתונים שיוצגו, עבור כל רשומה ורשומה, על ידי הגדרה מאפייני הפקדים.

לדוגמה, יהיה עליך להציב את תוכן השדה במאפיין Text של פקד TextBox או במאפיין Caption של פקד Label. כאשר משתמשים בפקדים באופן כזה, הם מכונים לרוב **פקדים בלתי מאוגדים** (Un-bound Controls). אחד היתרונות בשימוש בפקדים אלה הוא שבעת יישום אופן פעולה כזה ניתן להשתמש בכל סוגי הפקדים לשם הצגת המידע ולא רק בפקדי איגוד נתונים שיועדו מראש לשימוש בשילוב עם פקד Data.

גישה לנתונים המאוחסנים בשדות מסד הנתונים

למידע המאוחסן בשדות מסד הנתונים ניתן לגשת על ידי שימוש באוסף (Collection), Fields של אובייקט Recordset, באחד מכמה אופנים. לדוגמה, כל אחד מהמבנים התחביריים שיפורטו להלן יכול לשמש לאחזור הערך המאוחסן בשדה בשם ThisField לתוך מערך רשומות בשם rsTest והצגת הערך בתיבת טקסט ששמה Text1:

- ניתן להשתמש במספר הסידורי של השדה, במסגרת האוסף Fields. לדוגמה, המשפט Text1.Text = rsTest.fields(0) יציג את תוכן השדה ThisField בתיבת הטקסט Text1, זאת בהנחה ש-ThisField הוא השדה הראשון באוסף Fields.
- ניתן להסתייע בשם השדה כדי לאחזר את ערכו מתוך האוסף Fields: Text1.Text = rsText.Fields("ThisField").

- ניתן לנצל את העובדה כי Fields הוא האוסף המהווה ברירת מחדל בעת שימוש באובייקט Recordset: Text1.Text = rsTest("ThisField")
- לחילופין, ניתן להשתמש גם בגירסה מקוצרת של השיטה הקודמת שהוצגה: Text1.Text = rsTest!ThisField.

הערה:

אם שם השדה כולל בתוכו רווחים, תוכל להקיף את השם כולו בסוגריים מרובעים, כמו בדוגמה הבאה: Text1.Text = rsTest![longer field name]

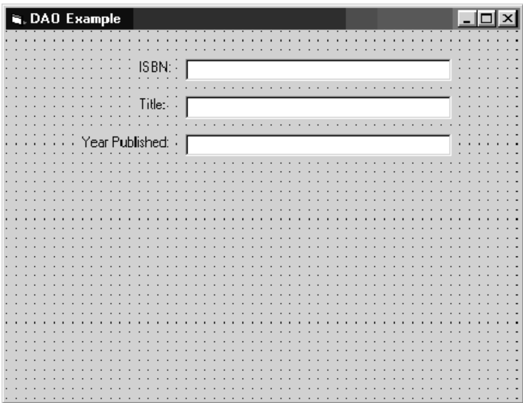
הצגת נתונים ביישום הדוגמה

כעת תוכל לשפר את יישום הדוגמה ולגרום לו להציג את הנתונים המאוחסנים במסד הנתונים. אם ביצעת את ההוראות עד כה, התוכנית שלך תיצור אובייקטי Database ו-Recordset על ידי כך שהיא תפתח את מסד הנתונים BIBLIO.MDB ואת הטבלה Titles שבו. המשך בפיתוח היישום על ידי כתיבת שגרות שתצגנה את הנתונים המאוחסנים בשדות ISBN, title ו-Year Published של הרשומה הפעילה, שאותן תכתוב על ידי ביצוע הפעולות שתפורטנה להלן:

1. הוסף לטופס שלושה פקדי TextBox וקרא להם txtISBN, txtTitle ו-txtYear.
2. הוסף פקדי Label שייזהו את פקדי TextBox.
3. הגדר את המאפיין Caption של כל פקד Label בהתאם לייעודו:

- ISBN:
- Title:
- Year Published:

לאחר הוספת פקדי TextBox ו-Label טופס היישום ייראה כבתרשים 26.3.



תרשים 26.3: שימוש בפקדים בלתי מאוגדים בשילוב עם אובייקטי גישה לנתונים לשם הצגת נתונים המאוחסנים במסד נתונים

4. הקלד את קוד User-define Sub Procedure המובא להלן, בחלון הקוד של הטופס frmDAOTest :

```
Sub ShowFields( )
	txtTitle.Text = rsTitles!Title
	txtISBN.Text = rsTitles!ISBN
	txtYear.Text = rsTitles![Year Published]
End Sub
```

5. הוסף את שורות הקוד המובאות להלן לסוף שגרת האירוע Load של הטופס :

```
rsTitles.MoveFirst
Call ShowFields
```

הערה:

עקב העובדה שהמאפיין Text הוא מאפיין ברירת המחדל של פקד תיבת טקסט, לא הכרחי לכלול את שמו בשורת הקוד המציבה בו ערך. אך מומלץ לכלול את שם המאפיין בשורת ההצבה, לשם שמירה על קריאות הקוד.

שינוי מיקום מצביע הרשומות

משום שמסד נתונים המכיל רק רשומה אחת אינו דבר מועיל במיוחד, מנגנון מסד הנתונים נדרש לכלול אמצעי כלשהו המאפשר מעבר בין הרשומות שבמערך. Visual Basic מציעה שש טכניקות לביצוע פעולות אלו :

טכניקה	תיאור
Move Methods	מעבירות את **מצביע הרשומות** (Record Pointer) מהרשומה הנוכחית לרשומה אחרת במערך
Find Methods	מאתרות את הרשומה הבאה העונה על תנאי החיפוש. פועלות על מערכי רשומות מסוג Dynaset ו-Snapshot
Seek Method	מאתרת את הרשומה הראשונה העונה על תנאי מסוים
Bookmark Property	מזהה מיקום רשומה מסוימת
AbsolutePosition	מעבירה את מצביע הרשומות למקום מסוים במערך
PercentPosition	מעבירה את מצביע הרשומות לרשומה הקרובה ביותר לשיעור הנתון מתוך גודלו הכולל של מערך הרשומות

לכל אחת מטכניקות אלו יש יתרונות וחסרונות, שיפורטו בסעיפים הבאים.

שימוש בשיטת Move

בקבוצת השיטות Move חמש שיטות שבהן ניתן להשתמש לצורך עבודה עם כל סוגי מערכי הרשומות הנתמכים על ידי Visual Basic :

שיטה	הפעולה המבוצעת
MoveFirst	מעבירה את המצביע מהרשומה הנוכחית לרשומה הראשונה
MoveNext	מעבירה את מצביע הרשומות מהרשומה הנוכחית לרשומה הבאה (זו שאחריה). אם לא קיימת רשומה כלשהי לאחר הנוכחית, **יופעל** הדגל End Of File (EOF) **ולא** תהיה רשומה נוכחית
MovePrevious	מעבירה את מצביע הרשומות מהרשומה הנוכחית לרשומה הקודמת לה. אם לא קיימת רשומה כזו, **יופעל** הדגל Beginning Of File (BOF) **ולא** תהיה רשומה נוכחית
MoveLast	מעבירה את מצביע הרשומות מהרשומה הנוכחית לרשומה האחרונה במערך
Move n	מעבירה את המצביע n רשומות קדימה מהרשומה הנוכחית (אם ערך n חיובי), **או** n רשומות אחורה מהרשומה הנוכחית (אם ערך n שלילי). אם פעולת ההעברה תעביר את מצביע הרשומות מעבר לגבולות המערך (EOF או BOF) תתרחש שגיאה. ניתן להעביר פרמטר נוסף שיהווה נקודת התחלה ממנה תחל הספירה

פקודות אלו תעברנה את מצביע הרשומות לרשומה הרצויה, תוך התייחסות לסדר בו ממוין כעת מערך הרשומות. סדר המיון הנוכחי של מערך רשומות הוא הסדר הפיסי שלו, למעט מקרים שבהם הופעל אינדקס על מערך רשומות מסוג Table או שבפקודת יצירת מערך מסוג Dynaset או snapshot נכללה הגדרת סדר מיון מסוים.

כעת הוסף לפרויקט קוד שידגים שימוש בשיטות MoveFirst, MovePrevious, MoveNext ו-MoveLast, על ידי ביצוע הפעולות המפורטות להלן :

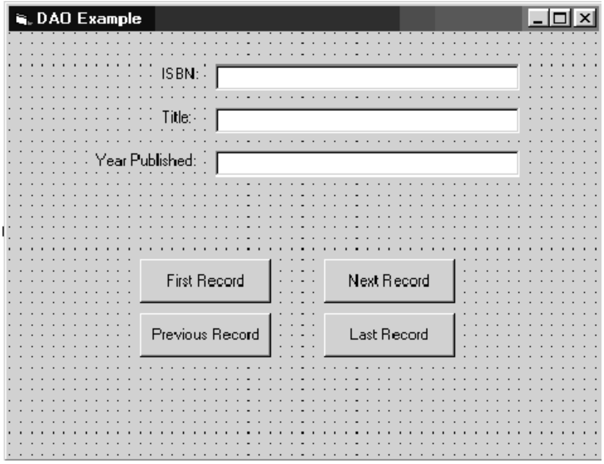
ראה, "הגדרת האינדקס הנוכחי בטבלה" בהמשך הפרק.

---

1. הוסף לטופס ארבעה פקדי Command Button וקרא להם cmdFirst, cmdPrevious, cmdNext ו-cmdLast.
2. הגדר את המאפיין Caption של כל אחד מהלחצנים כמפורט להלן :

- First Record
- Next Record
- Previous Record
- Last Record

בתרשים 26.4 תוכל לראות כיצד ייראה טופס היישום לאחר הוספת הלחצנים:



תרשים 26.4: הוסף לטופס לחצני פקודה שיאפשרו למשתמש לנווט במערך הרשומות

3. הוסף את שורות הקוד שלהלן לשגרת האירוע Click של הלחצן cmdFirst:

```
rsTitles.MoveFirst
Call ShowFields
```

4. הוסף את שורות הקוד שלהלן לשגרת האירוע Click של הלחצן cmdPrevious:

```
rsTitles.MovePrevious
Call ShowFields
```

5. הוסף את שורות הקוד שלהלן לשגרת האירוע Click של הלחצן cmdNext:

```
rsTitles.MoveNext
Call ShowFields
```

6. הוסף את שורות הקוד שלהלן לשגרת האירוע Click של הלחצן cmdLast:

```
rsTitles.MoveLast
Call ShowFields
```

כאשר תפעיל את הפרויקט, יוצגו ארבעה לחצני פקודה שיאפשרו לנווט במערך.

השיטה האחרונה שפורטה בקבוצת שיטות Move, Move n, מאפשרת להעביר את המצביע לרשומה שאינה בהכרח סמוכה לרשימה הנוכחית. המשתנה n יכיל את מספר הרשומות להעברת המצביע. המשתנה יכול להכיל ערכים חיוביים או ערכים שליליים, ערכים חיוביים משמשים כדי לנוע קדימה, וערכים שליליים לשם תנועה אחורה. להלן דוגמה לשימוש בשיטה להזזת המצביע שתי רשומות קדימה מהרשומה הנוכחית:

```
rsTitles.Move 2
```

שימוש במאפיין Bookmark

במצבים מסוימים תרצה שתהיה לך אפשרות לשוב לרשומה מסוימת לאחר הזזת מצביע הרשומה, או לאחר הוספת רשומות חדשות למערך. תוכל לעשות זאת על ידי שימוש במאפיין Bookmark של אובייקט Recordset. המאפיין Bookmark מכיל את ערך משתנה מחרוזת התואם לרשומה ושערכו הוא ייחודי עבור כל רשומה שבמערך. כדי שיתאפשר להשתמש במאפיין Bookmark תידרש בסך הכל ליצור משתנה מחרוזת "שיזכור" את ערך המאפיין Bookmark של אובייקט Recordset, לפני שתזיז את המצביע. כאשר תרצה להחזיר את המצביע לרשומה המקורית, יהיה עליך להציב את ערך אותו משתנה במאפיין Bookmark. בסעיף הבא נדגים ביצוע פעולה כזאת תוך שימוש בסדרת שיטות Find.

אזהרה:

אם תעבוד עם מסד נתונים מסוג השונה ממסד Jet, תידרש לבדוק את הערך המוצב **במאפיין** Bookmarkable של **אובייקט** Recordset **שבו** תשתמש, כדי לוודא שסוג מסד הנתונים שבו תבחר להשתמש תומך בשימוש ב-Bookmarks.

שימוש בשיטה Find

השיטות מסדרת Find ניתנות לשימוש רק בעת עבודה עם מערכי רשומות מהסוגים Dynaset ו-Snapshot (עקב העובדה שיישום הדוגמה נוצר תוך שימוש במערך רשומות מסוג Table לא יתאפשר להשתמש במסגרתו בשיטות כאלו במסגרתו). שיטות Find משמשות לאיתור רשומות אשר עונות על קריטריונים מסוימים. את הקריטריונים מנסחים תוך שימוש במבנה המשמש משפטי Where בשפת SQL, למעט שימוש במילת המפתח Where. ארבע שיטות Find המוגדרות ב- Visual Basic הן :

שיטה	הפעולה המבוצעת
FindFirst	מאתרת את הרשומה הראשונה העונה על סדרת הקריטריונים המפורטת בפקודה, תוך התחלת החיפוש מתחילת מערך הרשומות
FindNext	מאתרת את הרשומה הבאה במערך העונה על הקריטריונים המפורטים בפקודה תוך תנועה **מעלה** מהרשומה הנוכחיתהרשומות
FindPrevious	מאתרת את הרשומה הבאה במערך העונה על הקריטריונים תוך תנועה **מטה** מהרשומה הנוכחית
FindLast	מאתרת את הרשומה האחרונה במסד הנתונים העונה על הקריטריונים, תוך התחלת החיפוש מהרשומה האחרונה במערך

לאחר שתשתמש בשיטת Find כלשהי, תידרש לבדוק את ערך המאפיין NoMatch. אם המאפיין יכיל ערך True סימן שלא אותרה רשומה שעונה על הקריטריונים שהוגדרו. אם המאפיין יכיל ערך False, תדע שהרשומה שעליה יצביע המצביע עונה על הקריטריונים שהוגדרו בפקודת החיפוש.

טיפ:

ייתכן שתרצה להגדיר Bookmark עבור הרשומה הנוכחית בטרם תפעיל את אחת משיטות Find. אם תעשה כך, תוכל לשוב לרשומה זו, אם הפקודה לא תאתר רשומה כלשהי העונה על סדרת הקריטריונים שהוגדרה.

שיטות Find פועלות על ידי כך שהן סורקות כל רשומה בנפרד, כדי לאתר רשומות העונות על מערך הקריטריונים שהוגדר. נקודת התחלת החיפוש והכיוון בו יתבצע תלויים בשיטה שבה תבחר להשתמש. ביצוע פעולת החיפוש עלול להימשך זמן רב, כתלות בגודל מסד הנתונים ובשיטה שבה תבחר להשתמש. מנגנון Jet יוכל לבצע אופטימיזציה של פעולת החיפוש, אם ביטויי החיפוש שהוגדרו כלולים באינדקס כלשהו. אם אתה מתכוון לבצע פעולות חיפוש רבות על טבלה מסוימת, כדאי שתגדיר אינדקס או אינדקסים שיתבססו על שדות הכלולים באותה טבלה.

אם שיטת Find תצליח בפעולתה, המצביע יועבר לרשומה חדשה. אם פעולתה תיכשל, מאפיין NoMatch של אובייקט Recordset יוגדר True והמצביע יישאר על הרשומה שעליה הצביע בטרם ביצוע הפעולה. אחד האופנים שבהם תוכל להסתייע במאפיין NoMatch הוא על ידי כתיבת משפט If שיבדוק את הערך המוצב במאפיין.

בתוכנית 26.1 יוצג לפניך שימוש בשיטה FindFirst לאיתור רשומות, תוך שימוש במערך קריטריונים שיוגדר על ידי המשתמש.

תוכנית 26.1: DAOTEST.VBP - שימוש בשיטה FindFirst לחיפוש רשומה

```
Private Sub cmdFindFirst_Click()
	Dim sCriteria As String
	Dim sBkmark As String

	'Build the criteria string
	sCriteria = "title like '*" & txtFind.Text & "*'"

	'Remember where we were in case Find fails
	sBkmark = rsTitles.Bookmark

	'Execute the Find
	rsTitles.FindFirst sCriteria

	'Check for success
	If rsTitles.NoMatch Then
		MsgBox "Record not found!"
		'If failed, return to last good record
		rsTitles.Bookmark = sBkmark
	End If

	'Fill the text boxes
	ShowFields

End Sub
```

טיפ:

במקרים רבים, יצירה חוזרת של מערך רשומות מסוג Dynaset שתבוצע תוך התבססות על מערכת קריטריוני חיפוש תהיה מהירה יותר משימוש בפעולת Find לעיבוד כל הרשומות התואמות לאותה מערכת קריטריונים. תוכל גם ליצור ממערך רשומות מקורי מערך רשומות משני, מסונן, תוך שימוש בקריטריוני החיפוש כבמסנן. השיטה בה הפעולה תבוצע במהירות הגבוהה ביותר, תלויה בגורמים כגון כמות הנתונים הכוללת, גודל רשומה וגורמים נוספים. נסה לבצע פעולות שונות על הנתונים שלך, כדי לוודא מהי הפעולה הכדאית במצב נתון.

השגיאה Cannot Bind

במקרים בהם תשתמש במשתנים כערכי ייחוס להשוואה, ייתכן שתיתקל בהודעות שגיאה בנוסח Cannot Bind Item בעת הרצת התוכנית. במצבים בהם השדה והמשתנה שביניהם תתבצע ההשוואה יכילו ערכי טקסט, תוכל להקיף את שם המשתנה בגרשיים, כמתואר בקטע הקוד הבא:

```
Dim sFindCrit As String, sFindStr As String
sFindStr = "Smith"
sFindCrit = "Lastname = ' "& FindStr & """
rsTest.FindFirst FindCrit
```

למען שמירה על קריאות הקוד, תוכל גם להגדיר קבוע שבו תציב את תו הגרש ולהשתמש בקבוע זה במסגרת הקוד שלך.

באותו אופן תוכל גם לתחום משתני תאריך בתווי #, כדי לאפשר השוואת ערכים המאוחסנים בהם עם ערכים המאוחסנים בשדות מסוג Date. לא תידרש להוסיף תווים כלשהם כדי לאפשר השוואת מספרים.

הגדרת האינדקס הנוכחי בטבלה

ניתן להשתמש באינדקס בשילוב עם טבלה כדי להגדיר סדר מסוים עבור הרשומות בטבלה, או כדי לאפשר עבודה עם השיטה Seek לשם איתור מהיר של רשומות מסוימות. כדי שאינדקס יהיה אפקטיבי, תידרש להציב במאפיין Index של הטבלה שם אינדקס כלשהו שהוגדר עבור טבלה זו. שורת הקוד הבאה מגדירה אינדקס נוכחי:

```
rsTest.Index = "NameIndex"
```

חובה להציב במאפיין שם אינדקס קיים אשר כלול באוסף מסוג אינדקס של טבלה נתונה. אם תציב במאפיין שם אינדקס שאינו קיים, תתרחש שגיאה. מידע על יצירת אינדקסים חדשים תוכל למצוא בסעיף "יצירת אינדקס חדש" שיובא בהמשך הפרק.

הערה:

יצירת אינדקס לא תשנה את סדר אחסון הרשומות בפועל, רק את הסדר שבו הן תאוחזרנה מהטבלה ואת הסדר שבו הן תיסרקנה במסגרת פעולות החיפוש.

שימוש בשיטה Seek

שיטת Seek היא הדרך המהירה ביותר לאיתור רשומה מסוימת בטבלה, אך שיטה זו היא גם המוגבלת ביותר להגדרת מיקום במסגרת מערך הרשומות.

- השיטה Seek ניתנת לשימוש רק בעת עבודה עם מערכי רשומות מסוג Table, לא ניתן להשתמש בה לעבודה עם מערכי רשומות מהסוגים Dynaset או Snapshot.
- בשיטה Seek ניתן להשתמש רק במצבים בהם קיים Index פעיל. הפרמטרים של השיטה חייבים להתאים לערכי שדות האינדקס בו הם משתמשים.
- השיטה תאתר רק את השדה הראשון שיתאים לערך מסוים של האינדקס. פניות נוספות לשיטה לא תאתרנה רשומות נוספות המתאימות לערך הרצוי.

אופן פעולת השיטה Seek

פנייה לשיטה Seek מורכבת מקריאה לשיטה, מאופרטור השוואה (Comparison Operator) ומערכים עבור שדות המפתח של הרשומה. אופרטור ההשוואה שבפניה לשיטה יכול להיות אחד מבין האופרטורים הבאים: <, <=, =, >=, > או <>. ערכי המפתח שישמשו לצורך השוואה, נדרשים להיות מאותם סוגים הכלולים באינדקס השולט על פעולת ההשוואה. אומנם אין הכרח לכלול בביטוי ערכי מפתח כמספר השדות הכלולים באינדקס, אך תידרש לכלול ערך מפתח בעבור כל שדה שבו תרצה לבצע חיפוש. חובה לפרט את ערכי השדות בהתאם לסדר בו השדות פורטו בהגדרת האינדקס וכאשר הם מופרדים זה מזה בפסיקים. תוכנית 26.2 תציג שתי דוגמאות לשימוש בשיטה Seek. החלק הראשון ידגים את השיטה תוך שימוש באינדקס המבוסס על שדה יחיד. החלק השני יציג שימוש באינדקס המבוסס על שני שדות.

תוכנית 26.2: DAOTEST.VBP שימוש בשיטה Seek לאיתור רשומה מסוימת בטבלה

```
Dim dbTest As Database, rsDealers As Recordset

Set dbTest = OpenDatabase("C:\MARKET\MARKET.MDB")
Set rsDealers = dbTest.OpenRecordset("Dealers", dbOpenTable)
'**********************
'* SEEKING A DEALER ID *
'**********************
' The IDIndex index is based on the DealerID field.
' Set the recordset's index to the ID index
rsDealers.Index = "IDIndex"
' Look for dealer number 9534
rsDealers.Seek "=", 9534

' Display information or "Not Found" message as appropriate
If rsDealers.NoMatch Then
        MsgBox "Not Found"
Else
        MsgBox rsDealers!LastName & ", " & rsDealers!FirstName
End If
```

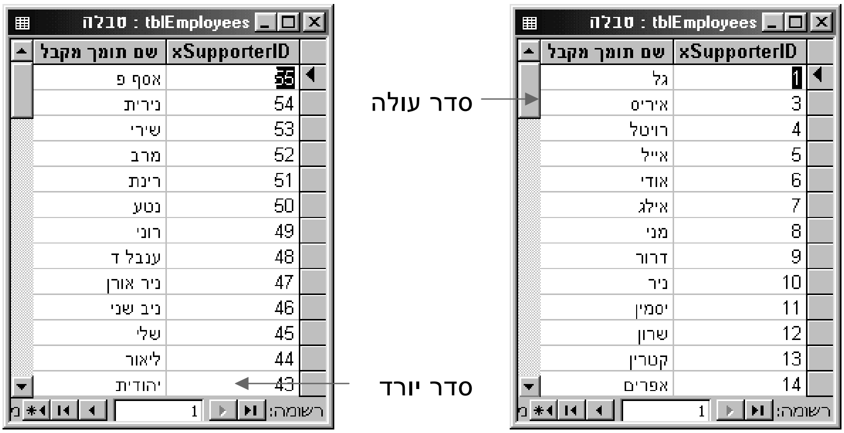
```
'***********************
'* SEEKING A DEALER NAME*
'***********************
' The NameIndex index is based on the
' LastName and FirstName fields.

' Set the recordset's index to the name index
rsDealers.Index = "NameIndex"

' Look for dealer named "Barbara Austin"
rsDealers.Seek "=", "Austin", "Barbara"

' Display information or "Not Found" message as appropriate
If rsDealers.NoMatch Then
        MsgBox "Not Found"
Else
        MsgBox rsDealers!LastName & ", " & rsDealers!FirstName
End If
```

קיים אופן פעולה אחד של השיטה Seek שלפני השימוש בו תידרש לבצע פעולות הכנה קפדניות. בעת שימוש באופרטורים המשווים =, >=, > או <> במסגרת שיטת Seek, החיפוש מתחיל ברשומה הראשונה הכלולה באינדקס הנוכחי ומתבצע תוך סריקת הרשומות לצורך איתור הרשומה הראשונה התואמת לערך החיפוש הרצוי. אם הפנייה לשיטה כוללת את האופרטור < או <=, פעולת החיפוש תתחיל ברשומה האחרונה של הטבלה ותתבצע תוך סריקה אחורה. אם האינדקס מכיל ערך ייחודי עבור כל רשומה, הסדר בו יתבצע החיפוש לא יהווה בעיה כלשהי. אך אם יש בשדות המפתח המפורטים בביטויי החיפוש ערכים החוזרים על-עצמם, הרשומה שתאותר תלויה באופרטור ההשוואה שבביטויי החיפוש ובסדר בו ממוין האינדקס. בתרשים 26.5 תוצג טבלה שמוינה תחילה על-פי שמות משפחה ואחר על-פי שמות פרטיים. הטבלה השמאלית בתרשים מוינה בסדר עולה ואילו הטבלה הימנית בתרשים מוינה בסדר יורד. בטבלה 26.4 תפורטנה פעולות Seek שונות שניתן לבצע על טבלה זו.



תרשים 26.5: ההבדלים בשימוש בסדר עולה במסגרת אינדקס לשימוש בסדר יורד

טבלה 26.4: השפעות השימוש באופרטורים משווים שונים ובסדרי מיון שונים של אינדקסים במסגרת פעולות Seek

אופרטור ההשוואה	סדר מיון האינדקס	הרשומה שתאותר
">=", "Smith, A"	עולה	Smith, Aaron
"<=", "Smith, Z"	עולה	Smith, Francis
">=", "Smith, A"	יורד	Schmidt, David
"<=", "Smith, Z"	יורד	Smith, Zeke

שימוש באופרטורים >, <, =< וכן => בשילוב עם אינדקס הממוין בסדר יורד דורש זהירות והקפדה. האופרטורים < וכן =< מאתרים רשומות שמקומן באינדקס הוא לאחר ערך המפתח הנתון בביטוי ההשוואה. זוהי הסיבה שבעטיה השימוש בביטוי החיפוש "Smith, A" ,">=" בשילוב עם אינדקס הממוין בסדר יורד, מחזיר את הרשומה המכילה את השם Schmidt, David. גם האופרטורים > וכן => פועלים באופן דומה. מהדוגמה שהובאה כאן הבנת בוודאי שיש להקפיד על בחירת האופרטור הנכון וסוג האינדקס הנכון שישמשו במסגרת פעולת Seek, כדי להבטיח שהפעולה תניב את התוצאות הרצויות לך.

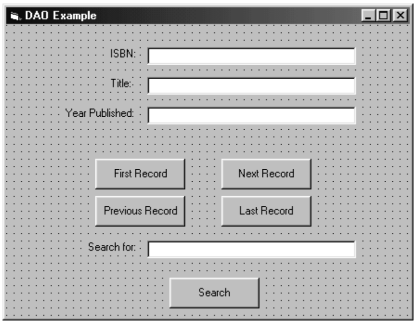
כפי שאירע בעת שימוש בסדרת השיטות Find, גם בעת שימוש בשיטה Seek, הצלחה באיתור רשומה התואמת לתנאי החיפוש, מעבירה את מצביע הרשומות לאותה רשומה. אם לא מאותרת רשומה התואמת לאותם תנאים, מוצב במאפיין NoMatch של אובייקט Recordset ערך False ומיקום מצביע הרשומות אינו משתנה.

הוספת פעולת seek ליישום הדוגמה

כעת תוכל לשפר את פרויקט הדוגמה על ידי כך שתוסיף לו יכולת לביצוע פעולות Seek. היות ואינך מעוניין שהמשתמשים יידרשו להקליד כותר ספר בשלמותו, כדי שיתאפשר להם לבצע חיפוש, תוכל להשתמש באופרטור =< כאופרטור ההשוואה שעליו תתבסס פעולת Seek. השימוש באופרטור זה ידרוש ממך לבדוק את תוכן הרשומה הנוכחית, גם במקרים בהם יוצב במאפיין NoMatch ערך False וזאת כדי לוודא שהרשומה אכן תואמת לתנאי החיפוש. בצע כעת את הפעולות הבאות:

1. הוסף לטופס פקד TextBox וקרא לו txtSearch.
2. הוסף לטופס פקד Label שייזהה את תיבת הטקסט והגדר את מאפיין Caption שלו: Search For.
3. הוסף פקד Command Button, הגדר את שמו cmdSeek והגדר את מאפיין Caption שלו: Search.

בתרשים 26.6 תוכל לראות כיצד ייראה הטופס לאחר הוספת הפקדים.



תרשים 26.6: הלחצן search מאפשר למשתמש להגדיר תנאי חיפוש

4. הוסף את שורות הקוד לשגרת האירוע Command_Click של הלחצן cmdSeek :

```
Dim sBkmark As Variant, sSeek As String
rsTitles.Index = "Title"
sBkmark = rsTitles.Bookmark
sSeek = UCase(txtSearch.txt)
rsTitles.Seek ">=", sSeek
If rsTitles.NoMatch Then
        MsgBox "No Match!"
        rsTitles.Bookmark = sBkmark
Else
        If UCase(Left(rsTitles!Title, Len(sSeek))) <> sSeek Then
                MsgBox "No match"
                rsTitles.Bookmark = sBkmark
        End If
End If
showFields
```

המאפיינים PercentPosition ו-AbsolutePosition

פרט למאפיין Bookmark, לאובייקט Recordset יש שני מאפיינים נוספים המשמשים להגדרת מיקום המצביע והם המאפיינים PercentPosition ו-AbsolutePosition.

המאפיין PercentPosition מגדיר מיקום יחסי של רשומה במערך. על ידי הצבת ערך שבין 0 ל-100 במאפיין, תגרום למצביע להצביע על הרשומה הקרובה ביותר לאותו מקום בקובץ. הצבת ערך שאינו בתחום שבין 0 ל-100 תגרום לשגיאה. במאפיין PercentPosition ניתן להשתמש לעבודה עם מערכי רשומות מכל שלושת הסוגים.

מאפיין AbsolutePosition מאפשר להעביר את מצביע הרשומות לרשומה מסוימת. הערכים שהוא יכול לקבל נעים בין 0 שייצג את הרשומה הראשונה, לבין ערך הקטן ב-1 ממספר הרשומות. הצבת ערך אחר, תגרום לשגיאה. כך שכדאי לכלול בדיקות לאיתור שגיאות בקטעי קוד. מאפיין AbsolutePosition מוגדר רק לאובייקטי Recordsets מסוג Dynaset ו-Snapshot. קטע הקוד הבא מדגים שימוש במאפיינים PercentPosition ו-AbsolutePosition. שים לב שמבוצעת בדיקת תקינות הערך המייצג את המיקום שאליו אתה מעוניין לעבור, הבדיקה מיועדת למנוע שגיאות.

```
'Move to the percent position specified
If fPct > 100 Then fPct = 100
If fPct < 0 Then fPct = 0
NewDyn.PercentPosition = fPct
        'Move to the absolute position specified
If lAbs > rsTest.RecordCount Then lAbs = NewDyn.RecordCount
If lAbs < 0 Then lAbs = 0
NewDyn.AbsolutePosition = lAbs
```

שימוש במסננים, אינדקסים ומיונים

Filters, Indexes ו-Sorts הם מאפיינים של אובייקט Recordset. ניתן להגדיר את המאפיינים האלה בעזרת פקודת הצבה כמו זו:

```
rsTest.Filter = "LastName = 'Smith'"
```

מסננים, אינדקסים ומיונים מאפשרים לך להגדיר תחום רשומות שיעובד ואת הסדר שבו תעובדנה הרשומות הכלולות בתחום. המאפיין Filter (מאפיין הסינון, הזמין לשימוש רק בעת עבודה עם מערכי רשומות מהסוגים Dynaset ו-Snapshot) מצמצם את מספר הרשומות שאליהן תתייחס פעולת העיבוד על ידי הגדרת קריטריונים שרק רשומות העומדות בהן תטופלנה, לדוגמה, רק רשומות ששמות המשפחה הכלולים בהן מתחילים באות M. Indexes (הזמינים לשימוש רק עם מערכי רשומות מסוג Table), ו-Sorts (הזמינים לשימוש רק עם מערכי רשומות מהסוגים Dynaset ו-Snapshot) מגדירים את סדר הצגת הרשומות במערך הרשומות, תוך התבססות על שדה או על שדות הכלולים באותו מערך רשומות. בעת השימוש ב-Sorts וב-Indexes ניתן גם להגדיר אם המיון יתבצע בסדר עולה או בסדר יורד.

הגדרת המאפיין Filter

המאפיין Filter זמין לשימוש אך ורק בעת עבודה עם מערכי רשומות מהסוגים Dynaset ו-Snapshot. אומנם התיאורים שיובאו בסעיף זה יתייחסו למערכי רשומות מסוג Dynaset, אך כל המידע שיובא יהיה נכון גם עבור מערכי רשומות מסוג Snapshot. הגדרת המאפיין Filter לא תשפיע על מערך הרשומות שאתו אתה עובד כעת, אלא תשמש לסינון רשומות שתועתקנה למערך רשומות אחר, שייווצר מהראשון.

תחביר הגדרת המאפיין Filter זהה לזה של משפט Where ב-SQL, פרט לכך שההגדרה אינה כוללת את מילת המפתח Where.

משפט הבחירה שבהגדרת המאפיין יכול להיות פשוט כגון 'State = 'TN או מורכב מכמה תנאים כגון 'State = 'AR' AND Lastname = 'Smith. ניתן לכלול בהגדרה גם ביטויים כגון '*Lastname LIKE 'S, שישמש לאיתור כל הרשומות שבהם שמות המשפחה מתחילים באות S. בסעיף "הגדרת מערך רשומות מסוג Dynaset", יש דוגמה לשימוש במאפיין Filter לצורך יצירת מערך רשומות חדש.

את התנאים שבביטויי Filter ניתן להגדיר גם באופן גמיש, באמצעות הכללת פונקציות. לדוגמה, לאיתור כל מדינות ארצות הברית, שהאות השנייה בקיצור שמן היא L, תוכל להשתמש בפונקציה Mid באופן הבא:

```
rsTest.Filter = "Mid(State,2,1) = 'L'"
```

השימוש בפונקציות אומנם מקנה לך גמישות, אך הוא גורם לחוסר יעילות בסינון הרשומות. את הסינון ניתן לבצע באופן יעיל יותר על ידי הכללת תנאי הסינון בשאילתה שתשמש ליצירת מערך הרשומות.

מידע נוסף אודות מסננים

הגדרת המאפיין Filter של מערך רשומות מסוג Dynaset לא תשפיע על מערך הרשומות שעליו אתה עובד כעת, אלא רק על מערכי רשומות שייגזרו ממנו. האופן היחיד שבו ניתן "לסנן" רשומות במערך רשומות קיים, הוא באמצעות שימוש בשיטות Find למעבר על פני הרשומות הכלולות בו. על ידי הצבת תנאי הסינון בתנאי החיפוש של שיטת Find, תוכל לעבד רק את הרשומות הרצויות לך.

במצבים שבהם תרצה לעבוד רק עם Dynaset "מסונן", כדאי שתבצע את הסינון על ידי הכללת משפט SQL מתאים במסגרת הקריאה לשיטה OpenRecordset שתשמש ליצירת מערך הרשומות.

הגדרת המאפיין Sort

בדומה למאפיין Filter גם המאפיין Sort זמין לשימוש רק בעת עבודה עם מערכי רשומות מהסוגים Dynaset ו-Snapshot. גם המידע שיובא בסעיף זה יתייחס רק למערכי רשומות מסוג Dynaset אך יהיה תקף גם עבור Snapshot. ניתן להגדיר את המאפיין Sort על ידי ציון שדות וסדר מיון (Ascending - עולה או Descending - יורד) שייושם על השדות שעל-פיהם ימוין מערך הרשומות. תוכל להשתמש לשם כך בכל שדה הכלול במערך הרשומות או בכל שילוב שדות הכלולים במערך הרשומות. תנאי Sort דומה במהותו לקטע ORDER BY הכלול במשפטי SQL. אופן השימוש במאפיין Sort במסגרת יצירת מערכי רשומות דומה מאוד לאופן השימוש במאפיין Filter, שתואר בסעיף הקודם. לדוגמה, שורת הקוד הבאה תגדיר שמערך הרשומות ימוין תחילה על-פי שם משפחה ואחר על-פי תאריך לידה:

```
rsTest.Sort = "LastName, Birthday"
```

אזהרה:
זכור שבעת הגדרת מפתח מיון המבוסס על כמה שדות, יש חשיבות לסדר בו יפורטו השדות. פעולת מיון שתבוצע תחילה על-פי שם פרטי ואחר כך על-פי שם משפחה תניב תוצאות שונות מפעולת מיון שתבוצע תחילה על-פי שם משפחה ואחר כך על-פי שם פרטי.

ניתן ליצור מערך רשומות ממוין, באופן יעיל יותר, על ידי הכללת קטע ORDER BY במשפט SQL שישמש ליצירת מערך הרשומות.

יצירת אינדקס חדש

בסעיף "הגדרת האינדקס הנוכחי טבלה" למדת כיצד להגדיר את האינדקס שישמש במסגרת פעולת חיפוש. אם האינדקס שבו תרצה להשתמש אינו קיים עדיין, תוכל להגדיר אותו על ידי כתיבת שורות קוד מעטות.

תחילה, יהיה עליך להיעזר בשיטה CreateIndex של אובייקט Table כדי ליצור אובייקט Index חדש. לאחר מכן יהיה עליך להוסיף לאינדקס שדות, על ידי שימוש בשיטה Append, כמו בדוגמה הבאה:

```
Dim Idx1 As Index
Dim Fld1 As Field
Set Idx1 = NewTbl.createIndex("Zip_Code")
Set Fld1 = Idx1.CreateField("Zip")
Idx1.FieldsAppend Fld1
```

קטע הקוד שהוצג כאן הגדיר אינדקס בשם "Zip Code". כדי שיתאפשר לך להשתמש באינדקס הזה נדרשת להוסיף אותו לאוסף Indexes של הטבלה ולהציב את שם האינדקס במאפיין Index של אובייקט Table:

```
NewTbl.Indexes.Append Idx1
NewTbl.Index = "Zip_Code"
```

אם לתוכנית שלך דרוש אינדקס מדוע שלא תגדיר אותו במסגרת עיצוב היישום, כך שלא תידרש לדאוג להגדיר אותו בזמן ריצה? ישנן כמה סיבות שבעטיין לא כדאי לך לנהוג כך:

- למנגנון מסד הנתונים נדרש זמן לשם עדכון האינדקס לאחר ביצוע פעולות להוספה, גריעה או שינוי רשומות. במצב בו מוגדרים אינדקסים רבים, התהליך הזה עלול לדרוש זמן רב. כך שייתכן שיהיה לך כדאי ליצור אינדקס רק כאשר הוא יידרש לך. זכור גם שאינדקסים צורכים משאבי דיסק, כך שהגדרת אינדקסים מיותרים עלולה לגרום לכך שליישום שלך יידרשו יותר משאבים מכפי שעומדים לרשותו.
- אתה מוגבל ל-32 אינדקסים לכל היותר במסגרת כל טבלה. זוהי כמות גדולה, אך אם יידרשו לך יותר מ-32 אינדקסים לצורך טיפול בטבלה מסוימת, תידרש ליצור חלק מהם רק כאשר הם יידרשו, ולגרוע אותם עם סיום פעולתם.
- ייתכן שתצליח לחזות מראש את כל האופנים שבהם היישום שלך יידרש להציג נתונים. אם תכלול ביישום שלך אמצעי שיאפשר הגדרת אינדקסים בזמן ריצה, תקנה למשתמשים ביישום גמישות רבה.

מבין הסיבות שפירטנו כעת, זו הזוכה לרוב למשקל הרב ביותר היא השפעת הצורך בעדכון אינדקסים רבים על ביצועי היישום. כדי שיתאפשר לך להחליט מה עדיף, הגדרת אינדקס בעת עיצוב היישום, או יצירת האינדקס רק כאשר הוא נדרש, כדאי שתפתח את היישום בשני האופנים, ותבדוק את ביצועי שתי הגרסאות.

הערה:

אומנם מומלץ להגביל את מספר האינדקסים הכלולים בטבלה, משיקולי שמירה על עדכניות, אך מאידך גם מומלץ להגדיר אינדקס לכל שדה המשמש בשכיחות גבוהה במסגרת שאילתות SQL. זאת מפני שמנגנון Jet (החל מגירסה 2.0 שלו) מיישם שיטה לאופטימיזציית שאילתות אשר מסתמכת על כל האינדקסים הזמינים לשימוש באותו רגע.

תוכניות המשנות רשומות מרובות

תוכניותמסוימות, או פונקציות הכלולות בתוכניות, מיועדות לאתר פריט מידע מסוים במסד נתונים. אך רוב התוכניות והפונקציות מיועדות לבצע פעולות על רשומות מרובות כקבוצה. יש שתי שיטות עיקריות לעבודה עם רשומות מרובות:

שיטה	הגדרה
לולאות	קבוצת פקודות שלה מבנה תחבירי מאחד הסוגים הבאים: Do...While, Do...Until או For...Next. הפקודות מבוצעות שוב ושוב, עד שמתקיים תנאי היציאה מהלולאה
משפטי SQL	פקודות הכתובות ב-SQL (Structured Query Language - שפת שאילתות מובנית), אשר מורות למנגנון מסד הנתונים לעבד רשומות. שפת SQL תתואר בפירוט בנספח 3

שימוש בלולאות

לולאות מהסוגים Do...While ו- For...Next הן בשימוש שכיח של רוב התוכניות. העקרונות המשמשים לעבודה עם לולאות כאלו תקפים גם בעת עבודה עם מערכי רשומות. כלומר שלולאת Do...While מתבצעת כל עוד תנאי מסוים מתקיים עבור מספר נתון רשומות. אופן נוסף המיושם לצורך עבודה עם רשומות מרובות יוצר **לולאות מרומזות** (Implied Loops). לרוב התוכנות אשר מבצעות פעולות להזנה ולהצגת נתונים יש בטפסים שלהם לחצני פקודה המאפשרים לעבור לרשומה הבאה או הקודמת. לחיצות חוזרות ונשנות על הלחצנים האלה גורמות לביצוע לולאות כאלו, על ידי הפעלה חוזרת ונשנית של אירועי Move. בעת הגדרת לולאות כאלו נידרש להקדיש מחשבה מיוחדת לטיפול במקרים מיוחדים כגון טיפול ברשומה הראשונה במערך הרשומות, ברשומה האחרונה או טיפול במקרה בו מערך הרשומות ריק. הבעיה שאתה נדרש להתמודד איתה במסגרת המקרים האלה היא שניסיון למעבר אחורה מהרשומה הראשונה, או למעבר קדימה מהרשומה האחרונה או לתנועה כלשהי במערך רשומות ריק, יגרום לשגיאה. אך למרבה המזל, מנגנון Jet מציע לך סיוע מסוים לטיפול במקרים כאלה. אובייקט Recordset כולל מספר מאפיינים אשר יתריעו בפניך על קיום תנאים אלה. המאפיינים האלה יתוארו בסעיף הבא.

תוכל להסתייע בארבעה מאפיינים חשובים של אובייקט Recordset במסגרת ביצוע פעולות לטיפול במערכי רשומות המכילים רשומות מרובות. טבלה 26.5 תפרט מעט אודות המאפיינים האלה.

טבלה 26.5: מאפיינים המשמשים לבקרה על לולאות לעיבוד רשומות

מאפיין	תיאור
BOF	הדגל Beginning-Of-File (תחילת קובץ), אשר מציין מצב בו מצביע הרשומות ממוקם לפני הרשומה הראשונה (מצב בו BOF=True) או לא (מצב בו EOF=True)
EOF	הדגל End-Of-File (סוף קובץ) אשר מציין מצב בו מצביע הרשומות ממוקם לאחר הרשומה האחרונה (EOF=True) או לא (EOF=False)
RecordCount	מכיל את מספר הרשומות שבמערך הרשומות שבוצעה אליהן גישה. המספר המאוחסן במאפיין מציין את מספר הרשומות שבמערך הרשומות רק בעקבות ביצוע גישה לרשומה האחרונה במערך הרשומות (למשל על ידי ביצוע פעולת MoveLast), אלא אם כן מערך הרשומות הנדון הוא מסוג Table
NoMatch	מציין שפעולת החיפוש האחרונה שבוצעה תוך שימוש ב-Find או Seek לא הצליחה לאתר רשומה התואמת לתנאי החיפוש

תוכל להיעזר במאפיינים האלה לצורך סיום פעולת הלולאות או לצורך מניעת מצבי שגיאה. נתייחס כעת לטופס הזנת הנתונים שהוצג בתרשים 26.3. כדי למנוע התרחשות שגיאה בעקבות לחיצה על הלחצן Next Record, כלול בתוכנית קטע קוד אשר יאפשר את המעבר רק בתנאי שמצביע הרשומות אינו מצביע על סוף הקובץ. קטע הקוד הבא יתייחס לאפשרות כזו:

```
If Not rsTitles.EOF Then
        rsTitles.MoveNext
        If rsTitles.EOF Then
                rsTitles.MoveLast
        End If
End If
```

לחילופין, תוכל לנטרל את הלחצן Next עם ההגעה לסוף הקובץ. ניתן ליישם את אותן שיטות גם עבור הלחצן Previous, אך תוך התייחסות לדגל BOF. ייתכן גם שתרצה לבדוק את הערך המאוחסן במאפיין RecordCount, כדי שבמצב שבו מערך הרשומות מכיל 0 רשומות, רק הלחצן Add Record יהיה פעיל.

הערה:

לאחר ביצוע פעולת MoveNext, ייתכן שמצביע הרשומות מצביע על סוף הקובץ (EOF). במצב כזה, לא קיימת רשומה נוכחית. כך שאם מצביע הרשומות אכן מצביע על סוף הקובץ, כדאי להשתמש בשיטה MoveLast כדי להבטיח שמצביע הרשומות יצביע על הרשומה האחרונה במערך הרשומות.

שימוש במשפטי SQL

בנוסף לשימוש בלולאות לשם עיבוד רשומות, תוכל להיעזר במשפטי SQL לשם ביצוע מספר פעולות המתייחסות לכמה רשומות בעת ובעונה אחת. הסעיפים הבאים ידונו בשני סוגים עיקריים של פונקציות כאלו:

- **שאילתות חישוב** (Calculation Queries) אשר מספקות נתונים מצטברים על קבוצות רשומות.
- **שאילתות פעולה** (Action Queries) שמוסיפות, גורעות או משנות קבוצות רשומות הכלולות במערך הרשומות.

מידע מפורט יותר אודות שפת SQL תוכל למצוא בנספח 3 **תקציר שפת SQL**.

שאילתות חישוב

שאילתות חישוב מאפשרות לחשב ערכים מצטברים המתייחסים לקבוצת רשומות, כגון: ממוצעים, ערכי מקסימום ומינימום ומספרי רשומות. שאילתות כאלו פועלות על ידי ביצוע פונקציה חישובית של SQL. ניתן גם להגדיר פילטר שישמש לסינון, על ידי שימוש במבנה WHERE סטנדרטי. בשאילתות כאלו תוכל להיעזר למשל כדי לחשב סך מכירות שבוצעו על ידי אנשי מכירות באזור מסוים, או שווי כולל של המלאי המצוי ברשותך נכון לתאריך מסוים (זאת כמובן בתנאי שהנתונים מצויים בטבלאות הבסיס).

נניח שמסד הנתונים שלך כולל טבלה המכילה נתונים אודות מכירת דגים. קטע הקוד המשמש לשם התייחסות לכל רשומה מאפשר לברר את סך המכירה ואת סוג הדג שנמכר במסגרתה. נתייחס כעת לקטע הקוד הבא:

```
sSQL = "SELECT SUM([Price]) As GrandTotal From FishSales WHERE FishCode = 1001"
Set rsTotal = dbFish.OpenRecordset(sSQL)
MsgBox "Grand Total = " & rsTotal("GrandTotal")
```

המפתח להבנת הדוגמה הוא הפונקציה SUM אשר מסכמת את הערכים שבשדה Price של כל אחת מהרשומות. מערך הרשומות שיוחזר על ידי הפונקציה יכיל שדה יחיד בשם GrandTotal אשר יכיל את הערך המחושב.

SQL כוללת כמה פונקציות נוספות, פרט לפונקציה SUM, אשר מחזירות ערכים מחושבים. לדוגמה, משפט SQL שלהלן, יחזיר את הערך המינימלי שבשדה Price של הקובץ, את הערך המקסימלי שבשדה זה ואת ממוצע הערכים המאוחסנים בשדה הזה.

```
SELECT MIN(Price) As MinPrice, AVG(Price) As AvgPrice, MAX(Price) As MaxPrice From _
    FishSales
```

שאילתת חישוב זו תיצור מערך Dynaset שיכיל רשומה יחידה שבה תשמרנה תוצאות החישובים. שימוש בשאילתת חישוב עשוי לחסוך שורות רבות, שתידרשנה לשם ביצוע פעולה זהה. בנוסף, שאילתות מתבצעות מהר יותר מקטעי קוד המשמשים לביצוע אותן פעולות, מפני שחישוביהם מבוצעים על ידי מנגנון מסד הנתונים במקום על ידי קוד Visual Basic.

שאילתות פעולה

שאילתות פעולה פועלות על מערך הרשומות באופן ישיר, לצורך הוספה, גריעה או שינוי קבוצת רשומות במערך. בדומה לשאילתות חישוב, גם הן מבצעות פעולות שבדרך אחרת תידרשנה שורות קוד רבות. להלן יפורטו כמה סוגי שאילתות פעולה:

- **עדכון ערכי שדות ברשומות** - פקודת UPDATE בשפת SQL יכולה לשמש לשם שינוי ערכים בשדות מרובים וברשומות מרובות, כמו בדוגמה הבאה:

```
UPDATE FishSales SET Price=Price*2 Where FishCode=1001
```

המשפט המוצג בדוגמה יכפיל פי שניים את הערך המאוחסן בשדה Price של רשומות שהערך בשדה קוד דג שווה ל-1001.

- **גריעת רשומות** - הפקודה DELETE מאפשרת לך לגרוע רשומות מטבלה.

```
DELETE From FishSales Where FishCode=1001 AND Price > 100
```

בעת שימוש בפקודה DELETE יש להיזהר, מפני שאם תשמיט בטעות את התנאי Where מהמשפט, תגרום למחיקת כל הרשומות שבמערך הרשומות.

- **הוספת רשומות** - הפקודה INSERT מאפשרת לך להגדיר ערכי שדות עבור רשומה חדשה. בשורת הקוד הבאה, ערכי השדות החדשים שנכללו בקטע ()Values של המשפט, פורטו על-פי סדר הופעת השדות בטבלת היעד:

```
INSERT INTO FishSales Values(1002, 19.95)
```

אם יש שדה מסוג AutoNumber, אין לכלול עבורו ערך בפקודת INSERT.

תוכל לבצע שאילתת פעולה, על ידי הכללת משפט SQL כפרמטר השיטה Execute של אובייקט Database, כמו בדוגמה הבאה:

```
dbFish.Execute "INSERT INTO FishSales Values(1003,  299.00)"
```

אם השאילתה שלך מיועדת לביצוע חוזר ונשנה, או שהיא כוללת פרמטרים, ייתכן שתרצה ליצור אובייקט QueryDef שיכיל את משפט SQL:

```
Dim NewQry As QueryDef
Set NewQry = dbFish.CreateQueryDef("MyQueryDef", "DELETE FROM FishSales")
NewQry.Execute
dbFish.DeleteQueryDef("MyQueryDef")
```

שים לב לעובדה ששאילתות פעולה אינן מחזירות רשומות, כך שלא תידרש לכלול אובייקטי Recordset במסגרת הקוד שלך. את הפעולות המבוצעות על ידי שאילתות פעולה ניתן לבצע גם באמצעות אחזור מערך רשומות ממסד הנתונים, ושימוש בשיטות כגון AddNew ו-Delete, אך ברוב המקרים שאילתות פעולה מבצעות פעולות כאלו באופן יעיל יותר, מפני שהן מבוצעות על ידי מנגנון מסד הנתונים.

הבנת פקודות תכנות נוספות

עד כה למדת בפרק זה כיצד לאתר רשומות מסוימות, וכיצד לעבור על פני קבוצות רשומות. אך רוב התוכניות תדרושנה ממך להוסיף, לשנות ולגרוע רשומות. הפקודות שאותן נסקור בסעיף זה תקפות רק לגבי מערכים מסוגי Table ו-Dynaset (זאת מפני שכפי שבוודאי זכור לך, מערכי רשומות מסוג Snapshot אינם ניתנים לעדכון).

הוספת רשומות

להוספת רשומות חדשות למערך, תיעזר בשיטה AddNew. השיטה אינה מוסיפה למעשה את הרשומה למערך, אלא רק מפנה את **מאגר ההעתקה** (Copy Buffer) כדי לאפשר את קליטת המידע הכלול ברשומה החדשה. לאחר שהשיטה הופעלה, התוכנית תוכל להציב ערכים בשדות הרשומה, על ידי כך שתתייחס אליהם כלמשתנים. לבסוף, יהיה עליך להשתמש בשיטה Update כדי להוסיף את הרשומה שבה הוצבו הערכים למסד הנתונים באופן פיסי. אופן פעולה זה יודגם בקטע הקוד הבא:

```
'Prepare a new record in the copy buffer
rsTest.AddNew

'Populate the new record's fields
rsTest!FirstName = "Casey"
rsTest!LastName = "Jones"

'"Save" the changes with update
rsTest.Update
```



אזהרה:

עקב העובדה שהשיטה AddNew מאחסנת נתונים במאגר ההעתקה, שימוש חוזר בשיטה או הזזת מצביע הרשומות תוך שימוש בשיטות Move או Find (בטרם בוצעה פנייה לשיטה Update) מפנה את המאגר. כך שביצוע פעולות כאלו מוחק את כל הנתונים שהוזנו עבור הרשומה החדשה.

עריכת רשומות

השיטה Edit משמשת לביצוע שינויים ברשומה, באופן דומה להוספת רשומה. גם שיטה זו מאחסנת עותק של הרשומה הקיימת במאגר ההעתקה, כדי לאפשר ביצוע שינויים בתוכן הרשומה. בדומה לשיטה AddNew, גם תוצאות פעולת Edit נכנסות לתוקף רק לאחר ביצוע פעולת Update.

כדי לאפשר עריכת רשומות במסגרת יישום, תידרש להציע למשתמשים אמצעי שיאפשר להם לאתר את הרשומה שהם מעוניינים לערוך. תוכל להשתמש לשם כך בשיטות השונות שתוארו בפרק זה. לאחר שתאתר את הרשומה הרצויה (ולאחר שתהפוך אותה לרשומה הנוכחית), יהיה עליך להפעיל את שיטת Edit. לאחר מכן תוכל להציב ערכים חדשים בשדות הרשומה (כאשר סביר להניח שהערכים החדשים יתקבלו

מהמשתמשים) ולבסוף תידרש להפעיל את השיטה Update. אופן פעולה זה יודגם במסגרת קטע הקוד הבא:

```
rsTest.Edit
rsTest!FirstName = txtFirstName.Text
rsTest!LastName = txtLastName.Text
rsTest.Update
```

אזהרה:

עקב העובדה שהשיטה Edit מאחסנת נתונים במאגר ההעתקה, שימוש חוזר בשיטה או הזזת מצביע הרשומות תוך שימוש בשיטות Move או Find (בטרם בוצעה פנייה לשיטה Update) מנקה את המאגר. כך שביצוע פעולות כאלו מוחק את כל הנתונים שהוזנו עבור הרשומה החדשה.

עדכון רשומות

בשני הסעיפים האחרונים ראית שכדי לבצע שינויים ברשומות, נידרש להשתמש בשיטה Update בשילוב עם השיטות AddNew ו-Edit. השיטה Update כותבת את הנתונים המאוחסנים במאגר ההעתקה, במערך הרשומות. בדומה לשיטה AddNew גם השיטה Update מוסיפה למערך הרשומות, רשומה ריקה שבה נרשמים הנתונים. בעת עבודה בסביבות מרובות משתמשים, השיטה Update היא גם זו שמבטלת את מצבי הנעילה שנגרמו על ידי ביצוע פעולות Add ו-Edit המצויות כעת במצב המתנה.

הערה:

במצבים בהם תיעזר בפקדי נתונים לשם עבודה עם מערכי רשומות, לא תידרש להשתמש בשיטה Update. השיטה מופעלת באופן אוטומטי בכל פעם שמבוצע מעבר לרשומה אחרת במערך הרשומות.

גריעת רשומות

גריעת רשומה ממערך רשומות, דורש שימוש בשיטה Delete. הפעלה השיטה Delete בעקבות איתור רשומה והפיכתה לרשומה הנוכחית גורעת את הרשומה מהטבלה, באופן קבוע. השיטה תגרע את הרשומה ממערך הרשומות ותציב ערך Null (ערך ריק) במצביע הרשומה. אופן השימוש בשיטה Delete מודגם על ידי שורת הקוד הבאה:

```
rsTest.Delete
```

אזהרה:

לאחר שתגרע רשומה מטבלה, היא תיעלם ללא אזהרה כלשהי. תוכל לשחזר אותה רק אם השתמשת בפקודה BeginTrans בטרם המחיקה. כך תוכל להשתמש ב-Rollback כדי לבטל את הטרנזקציה. אם לא השתמשת ב-BeginTrans, תוכל לשחזר את הרשומה על ידי יצירה מחדש, תוך שימוש ב-AddNew.

הערה:

פעולות גריעה ושינוי רשומות מבוצעות מבלי שהמשתמש מתבקש לאשר את ביצוען. אם תרצה שהתוכנית תבקש מהמשתמש אישורים לביצוע פעולות כאלו, תידרש לכתוב את הקוד הדרוש לשם כך בעצמך. הדרך הפשוטה ביותר לעשות זאת היא על ידי שימוש בפונקציה MsgBox. השימוש בפונקציה זו יאפשר לך להזהיר את המשתמשים מפני הפעולה העומדת להיות מבוצעת ולבקש מהם לאשר את ביצועה.

מבוא לעיבוד טרנזקציות

עיבוד טרנזקציות (Transaction Processing) מאפשר לך להתייחס לסדרת פעולות לשינוי, הוספה וגריעת רשומות כליישות אחת. זוהי יכולת מועילה במצבים בהם שינוי מסוים האמור להתבצע במסד הנתונים תלוי בשינויים אחרים, ושאתה מעוניין לוודא שכל השינויים בוצעו כנדרש, לפני שיהפכו לקבועים. נניח שיש לך יישום המטפל בנקודות מכירה, אשר מעדכן את נתוני המלאי בעקבות ביצוע מכירות. בכל מקרה הזנת פריט במסגרת טרנזקציית המכירות, מבוצע שינוי במסד נתוני המלאי. אך אתה מעוניין שהשינויים בנתוני המלאי יישמרו רק עבור פעולות מכירה שהושלמו. במידה שפעולת המכירה לא תושלם, תרצה להחזיר את מסד נתוני המלאי למצבו ההתחלתי. עיבוד טרנזקציות היא פונקציה של האובייקט Workspace, כלומר שהיא תשפיע על כל מסדי הנתונים שיהיו פתוחים בעת ביצועה.

Visual Basic מציעה לך שלוש שיטות המאפשרות ביצוע פעולות לעיבוד טרנזקציות. שיטות אלו מבצעות את הפעולות שתפורטנה מייד:

שיטה	תיאור
BeginTrans	מתחילה טרנזקציה ומגדירה את המצב ההתחלתי של מסד הנתונים
RollBack	מחזירה את מסד הנתונים למצב בו היה בטרם ביצוע פקודת BeginTrans. ביצוע הפעולה מבטל את כל השינויים שבוצעו מאז פקודת BeginTrans האחרונה שבוצעה
CommitTrans	שומרת את כל השינויים במסד הנתונים שבוצעו מאז פקודת beginTrans האחרונה שניתנה. לאחר מתן פקודת CommitTrans לא ניתן לבטל פעולות שבוצעו במסגרת הטרנזקציה

תוכנית 26.3 תציג דוגמה לשימוש בפקודות BeginTrans, RollBack ו-CommitTrans במסגרת תוכנית. השימוש בטרנזקציה נעשה כדי לאפשר ביטול הזמנה כל עוד לא הושלם העיבוד שלה.

תוכנית 26.3: TRANSACT.TXT - שימוש בעיבוד טרנזקציות כדי לאפשר התייחסות לשינויים שבוצעו במסד נתונים כלמקשה אחת

```
OldWs.BeginTrans
'*********************************************
' Perform loop until user ends sales transaction
'*********************************************
Do While Sales
'*********************************************
' Get item number and sales quantity from form
' Input Itemno,SalesQty
' Ind item number in inventory
'*********************************************
        Inv.FindFirst "ItemNum = " & Itemno
'*************************
' Update inventory quantity
'*************************
        Inv.Edit
        Inv("Quantity") = Inv("Quantity") - SalesQty
        Inv.Update
Loop
'*****************************************
' User either completes or cancels the sale
'*****************************************
If SaleComp Then
        OldWs.CommitTrans
Else
        OldWs.Rollback
End If
```

מכאן...

פרק זה הציג בפניך את נושא אובייקטי גישה לנתונים, המהווים אמצעי לעבודה עם מסדי נתונים מתוך Visual Basic. הבנה של טכנולוגיית DAO תקנה לך יסוד טוב להבנת טכנולוגיות אחרות לגישה לנתונים כגון RDO ו-ADO. הדרך הטובה ביותר ללמוד דברים נוספים אודות טכנולוגיית DAO היא על ידי ניסיון ליישם את התכנים שנלמדו בפרק זה, תוך שימוש במסדי הנתונים לדוגמה שסופקו יחד עם Visual Basic, כדוגמת BIBLIO. למשתמשים בלתי מנוסים הייתי ממליץ לשמור עותקי גיבוי של מסדי הנתונים האלה, כדי שלא יחששו לבצע פעולות במסגרתם.

חלק מהנושאים שהוזכרו בפרק זה מוצגים בפירוט רב יותר בחלקים אחרים בספר זה. כדאי שתעיין בפרקים הבאים:

- תמצית המושגים הבסיסיים הקשורים בעבודה עם מסדי נתונים תוכל למצוא בפרק 24 **יסודות מסד הנתונים**.
- אם תרצה ללמוד אודות ממשקי גישה לנתונים המיועדים לשימוש במסגרת מסדי נתונים בנויים במבנה שרת/לקוח, עיין בפרק 27 **גישה לאובייקטי נתונים מרוחקים (RDO)**.
- מידע נוסף אודות שיטת הגישה לנתונים שתתפוס את מקום ADO ו-RDO תוכל למצוא בפרק 28 **גישה לאובייקטי נתונים באמצעות פקדי ActiveX**.
- שפת SQL תתואר בפירוט נרחב יותר בנספח 3 **תקציר פקודות SQL**.

27

גישה לאובייקטי נתונים מרוחקים (RDO)

מה בפרק?

- תפיסות לגבי גישה לנתונים
- עבודה עם ODBC
- אובייקטי גישה לנתונים מרוחקים
- שימוש בפקד Remote Data

בדיון שלנו בנושא הגישה למסדי נתונים התמקדנו עד כה במסדי נתונים המבוססים על מחשבי PC. כלומר התייחסנו למסדי נתונים כגון Access, FoxPro, dBase ו-Paradox. אך Visual Basic היא גם כלי טוב מאוד ליצירת יישומי חזית (Front-Ends) נוספים המיועדים לפעול בסביבות שרת/לקוח (Client/Server). יישומי חזית משמשים לשם גישה למסדי נתונים המאוחסנים בשרתים כגון SQL Server או Oracle. רוב הפעולות הקשורות לפיתוח יישומי חזית, כגון עיצוב טפסים וכתיבת קוד לעיבוד מידע - תהיינה זהות בעת פיתוח מסדי נתונים מבוססי PC, ובמסגרת כתיבת יישומי שרת/לקוח. ההבדל העיקרי בין שני סוגי היישומים הוא בתחום ההתקשרות לנתונים.

פרק זה ייראה לך כיצד לאפשר לתוכניות הכתובות ב- Visual Basic גישה קלה לנתונים המאוחסנים במחשבים מרוחקים, על ידי שימוש באובייקטי גישה לנתונים מרוחקים (RDO). במהלך הפרק תלמד שהוספת Remote Data Objects ל״מאגר״ כלי הפיתוח שלך תוכל להקל עליך את העבודה עם נתונים המאוחסנים במחשבים מרוחקים.

תפיסות לגבי גישה למסדי נתונים

בטרם נשקיע עצמנו בנושא הגדרת יישומים שיבצעו גישה למסדי נתונים בעלי מבנה שרת/לקוח, נבחן את הבדלים בפילוסופיות של שני סוגי מסדי הנתונים. בעולם מסדי הנתונים המבוססים על מחשבי PC, הגישה לנתונים מתבצעת תוך שימוש ב**מנגנון מסד נתונים** (Database Engine), המהווה חלק מהיישום. ב- Visual Basic מנוע Jet מהווה חלק מהיישום ומנהל את מסד הנתונים. כאשר משתמש נותן הוראות המיועדות לאחזר נתונים ממסד הנתונים, הן מתורגמות על ידי המנגנון, והעיבוד מתבצע על המחשב המקומי. המנגנון נותר על המחשב שלך, הן אם מסד הנתונים הוא מקומי והן אם הוא מרוחק. היישום הוא שמכיל את הלוגיקה המאפשרת לו גישה ישירה לקבצי המידע של מסד הנתונים. יישומי שרת/לקוח פועלים באופן שונה. היישום מעביר דרישה לקבלת מידע, לרוב בצורת משפט SQL. הדרישה מועברת לשרת מסד הנתונים (Database Server), אשר מעבד אותה ומחזיר את תוצאות העיבוד. זוהי פעולת שרת/לקוח אמיתית, שבמסגרתה השרת הוא שמעבד את הדרישה.

למערכות שרת/לקוח יש מספר יתרונות על פני שימוש משותף במסד נתונים. ראשית, הלוגיקה שלהם מתנהלת באתר מרכזי, המקל על תחזוקתם. לדוגמה, נניח שיש לך מערכת המבצעת חישובי מס קניה תוך הסתמכות על כללים שהוגדרו על ידי חברתך. במערכת שרת/לקוח, הלוגיקה החישובית תהיה מאוחסנת בשרת מסד הנתונים. האחסון המרכזי מאפשר לבצע את כל השינויים הדרושים במקום אחד, מבלי להידרש לשכתב יישומי לקוח. יתרונות נוספים של שיטת שרת/לקוח הם היכולת לבזר עיבוד והיכולת להפריד בין הלוגיקה העסקית לבין ממשק המשתמש.

עבודה עם ODBC

אחת השיטות המיושמות על ידי Visual Basic לתקשורת עם מסדי נתונים המתנהלים בשיטת שרת/לקוח קרויה **ODBC - Open Database Connectivity** (קישוריות פתוחה למסדי נתונים). ODBC היא רכיב בארכיטקטורת **WOSA** - **Windows Open System Architecture** (ארכיטקטורת מערכות פתוחות מבוססת Windows). מנגנון ODBC מספק את אוסף הפונקציות API - Application Programming Interface (ממשק תכנות יישומים), המקל על המפתח להתקשר למסדי נתונים מסוגים שונים. ODBC מאפשר לך להשתמש באותה סדרת פקודות לשם גישה לנתונים המאוחסנים בשרתי מסדי נתונים מסוגים שונים, כגון SQL Server, Oracle או Interbase וזאת על-אף ההבדלים באופני אחסון הנתונים המיושמים במסגרת מערכות אלו. פונקציות ODBC גם מאפשרות לך גישה למספר סוגים שונים של מסדי נתונים מבוססי PC.

הבנת מנהלי התקן ODBC

מנהלי התקן ODBC (ODBC Drivers) הם קבצי DLL המכילים פונקציות המאפשרות להתקשר למסדי נתונים שונים. לכל סוג מסד נתונים מוצע מנהל התקן נפרד. מנהלי ההתקן המיועדים לשימוש עם מסדי נתונים סטנדרטיים, כגון מסדי נתונים מבוססי PC או SQL Server מסופקים עם Visual Basic. מנהלי התקן למסדי נתונים אחרים מסופקים על ידי היצרנים של שרתי מסדי הנתונים.

הערה:

אם תשתמש ב-ODBC ביישום שתפתח, יהיה עליך להקפיד להפיץ את מנהלי ההתקן הדרושים יחד עם היישומים. אם בחרת את האופציה ODBC Redistributable בהתקנת Visual Basic, סביר להניח שנוצרה תיקיית משנה בשם ODBC בתיקיית Visual Basic. אופציה זו גורמת לתוכנית ההתקנה של Visual Basic להתקין את מנהלי ההתקן של ODBC, עדיין יהיה עליך להתקין מקורות נתונים (Data Sources) בנפרד.

ל-ODBC שני סוגי מנהלי ההתקן : **Single-Tier** או **Multi-Tier**. הסוג הראשון משמש להתקשרות למסד נתונים מבוסס PC המאוחסן על מחשב מקומי או על שרת קבצים. הסוג השני משמש להתקשרות למסדי נתונים הפועלים בשיטת שרת/לקוח, בה מתבצע עיבוד משפטי SQL על ידי השרת ולא על ידי המחשב המקומי.

כל מנהל התקן של ODBC מכיל מערכת פונקציות בסיסית, הידועה כ- **Core-level Capabilities** (יכולות רמת הליבה). יכולות אלו כוללות :

- תמיכה בהתקשרות למסדי נתונים
- הכנה ועיבוד משפטי SQL.
- עיבוד טרנזקציות.
- החזרת התוצאות של פעולות העיבוד.
- יידוע יישומים על שגיאות.

הגדרת מקורות הנתונים של ODBC

בטרם תוכל להשתמש ב-ODBC להתקשרות למסד נתונים, תידרש לוודא שני דברים :

- שבמערכת שלך מותקנים מנהלי התקן של ODBC.
- שהגדרת **מקור נתונים** (Data Source) של ODBC.

את שתי הפעולות תוכל לבצע בעזרת היישום ODBC Manager. את הפעולה השנייה תוכל לבצע גם מקוד תוכנית, תוך שימוש באובייקטי גישה לנתונים. זכור שמנהל התקן ODBC משמש לשם התקשרות למסד נתונים **מסוג מסוים**, כגון SQL Server. מקור נתונים של ODBC הוא תצורה המוגדרת עבור מנהל התקן ODBC, המשמשת לגישה ל**מסד נתונים ספציפי**, כגון מסד נתונים להנהלת חשבונות החברה שלך.

הערה:

במערכות ההפעלה Windows 95 **ו-** Windows 98, תמצא את היישום ODBC Manager בלוח הבקרה, ניתן לגשת אליו על ידי בחירה ב**הגדרות** מתפריט **התחלה**. הסמל שאותו אתה מחפש מסומן בתווית **32-bit ODBC**. ייתכן שתמצא גם את הסמל המסומן בתווית **ODBC**, אם מותקנים אצלך יישומים ישנים בני 16Bit.

הערה:

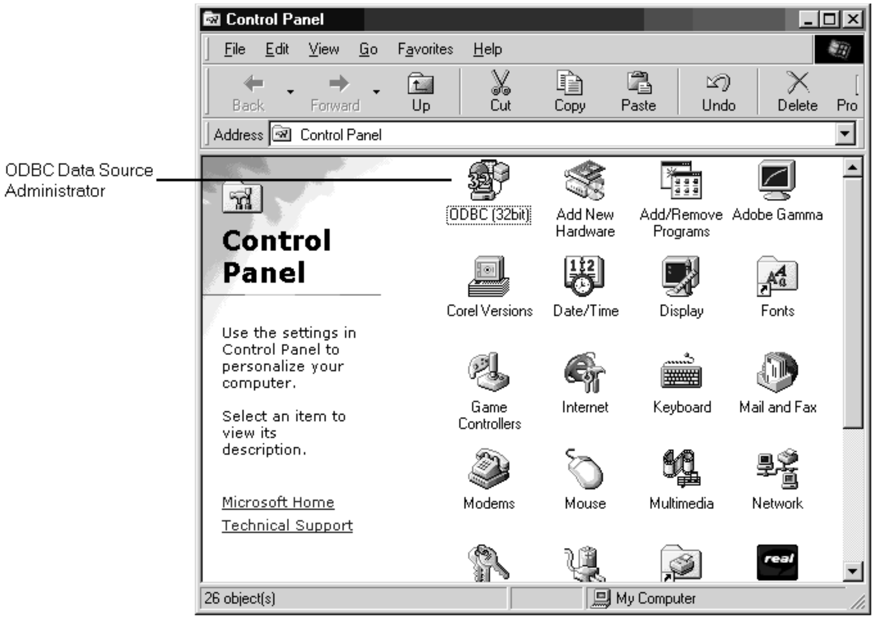
כדי להבטיח שכל הקוראים יוכלו להיעזר במידע המובא בפרק זה, השתמשנו במנהל ההתקן ODBC Access במסגרת כל הדוגמאות שבפרק. Access הוא אומנם מסד נתונים מבוסס PC, אך את השיטות לגישה אליו ניתן ליישם גם לגבי פנייה למסדי נתונים הפועלים על שרתים. חשוב שתזכור שהתקשרות לקובץ Access.MDB תוך שימוש ב-ODBC שונה מהתקשרות ישירה תוך שימוש במנגנון Jet.

גישה למנהלי ההתקן של ODBC

להתקנת מקורות הנתונים של ODBC במערכת שלך, תידרש להשתמש ביישום ODBC Windows Data Source Administrator. את היישום תמצא בלוח הבקרה כמוצג בתרשים 27.1.

תיבת הדו-שיח ODBC Administrator

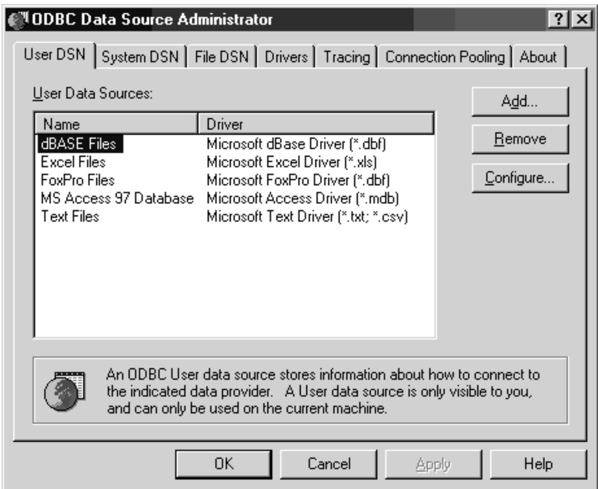
אם תיבות הדו-שיח שיוצגו לפניך בעקבות לחיצה על הסמל 32-bit ODBC תהיינה שונות מאלו הנראות בתרשימים, אל תיבהל. בדומה לרוב מוצריה, Microsoft שיווקה גם כמה גרסאות שונות של ODBC. ODBC נכלל במוצרים שונים ביניהם Office ו- Visual Basic. ייתכן שהגירסה המותקנת אצלך היא העדכנית ביותר אך ייתכן גם שלא (הגירסה תלויה באפשרויות שבחרת במסגרת ההתקנה). גרסאות ישנות יותר אינן כוללות תיבות דו-שיח הבנויות מכרטיסיות. במקרה הצורך, זכור כי ODBC סופק יחד עם Visual Basic, כדי לאפשר לך לעדכן את ההתקנות שברשותך.



תרשים 27.1: ליישום ODBC Data Source Administrator ניתן לגשת על ידי לחיצה על הסמל ODBC 32 bit שבלוח הבקרה

כאשר תפתח את היישום ODBC Manager, תוצג לפניך תיבת הדו-שיח ODBC Data Source Administrator, המוצגת בתרשים 27.2. תיבת דו-שיח זו כוללת כמה כרטיסיות המאפשרות לך להוסיף מקורות נתונים ולהנהל מנהלי התקן נוספים.

ייתכן ששמת לב לכך שכותרות שלוש הכרטיסיות הראשונות בתיבת הדו-שיח מסתיימות ב-DSN. **DSN** הם ראשי תיבות Data Source Name (שם מקור נתונים). DSN הוא המפתח המשמש את התוכנית שלך לשם זיהוי מקור נתוני ODBC. ODBC מטפל עבורך בקישור DSN למנהל ההתקן, השרת וקובץ מסד הנתונים.



תרשים 27.2: תיבת הדו-שיח Data Source Administrator מאפשרת להגדיר את מקורות הנתונים

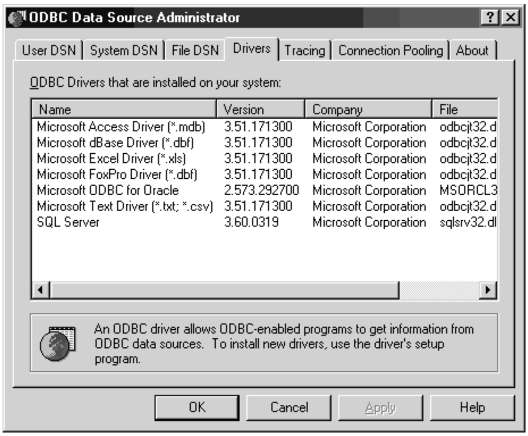
מקורות הנתונים של ODBC נחלקים לשלושה סוגים: User, System ו-File. מטרת כל סוגי DSN זהה - אספקת מידע על מקור נתונים מסוים - אך ישנם הבדלים בנסיבות ובעיתוי שבהם מתאפשר לך להשתמש בכל אחד מהסוגים:

- System DSN, ישים יותר בסביבת Windows NT מאשר ב- Windows 95/98, ואינו מקושר לפרופיל משתמש ספציפי. מכך משתמע שלאחר הגדרת פרופיל כזה מתאפשר לכל התוכניות והשירותים הפועלים על אותו מחשב, לגשת אליו. לדוגמה, אם אתה משתמש Internet Information Server לצורך התקשרות למסד נתונים, סביר להניח שתגדיר עבור מסד נתונים זה DSN מסוג System.
- File DSN מאחסן את נתוני DSN בקובץ טקסט. קובץ טקסט זה הוא קובץ INI המכיל נתונים על מנהל ההתקן של מסד הנתונים ועל מיקום מסד הנתונים. קובץ DSN אינו מקושר עם מחשב מסוים, כך שניתן לאחסן אותו בדיסק רשת.
- User DSN הוא הסוג שסביר להניח שתרבה בשימוש בו. נתוני User DSN מאוחסנים ברישום המערכת במחשב המקומי. במערכת Windows NT כל User DSN מקושר עם פרופיל משתמש מסוים ואינו מוכר מחוץ לאותו פרופיל.

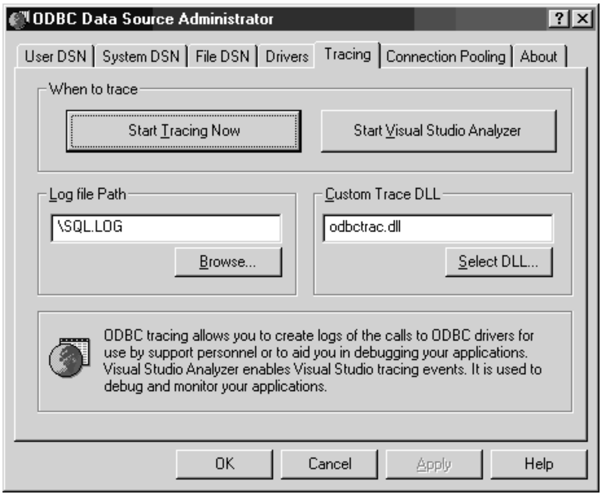
שלוש הכרטיסיות הנותרות בתיבת הדו-שיח ODBC Source Administrator משמשות לצרכים אינפורמטיביים ולניפוי שגיאות. בכרטיסיה ODBC Drivers, המוצגת בתרשים 27.3, מוצגת רשימת מנהלי התקן ODBC המותקנים במערכת שלך.

הכרטיסיה האחרונה, About, דומה לכרטיסיה ODBC Drivers. בכרטיסיה מפורטים מספרי גרסאות הקבצים המשמשים את ODBC עצמו. שתי הכרטיסיות עשויות לסייע לך בבדיקה אם אצל משתמשי היישום שלך מותקנים מנהלי ההתקנים הדרושים.

לפני שנעבור הלאה כדאי שנתייחס לכרטיסיה Tracing המוצגת בתרשים 27.4. כרטיסיה זו מאפשרת לך לעקוב אחר קריאות למנהלי ההתקן של ODBC אשר מבוצעות על ידי ODBC Manager. זכור כי ODBC הוא אמצעי המאפשר לך להתקשר למסדי נתונים שונים תוך שימוש בסדרת פונקציות API משותפות. הכרטיסיה Tracing תאפשר לך לבצע מעקב אחר קריאות כאלו. סביר להניח כי לא תרבה בביצוע פעולות מעקב כאלו, אך הן עדיין בחזקת דבר שכדאי לדעת על קיומו.



תרשים 27.3: הכרטיסיה ODBC Drivers מודיעה אילו מנהלי התקן מותקנים במערכת



תרשים 27.4: הכרטיסיה ODBC Administrator Tracing היא כלי עזר לניפוי שגיאות

יצירת מקור נתונים ODBC בעזרת ODBC Manager

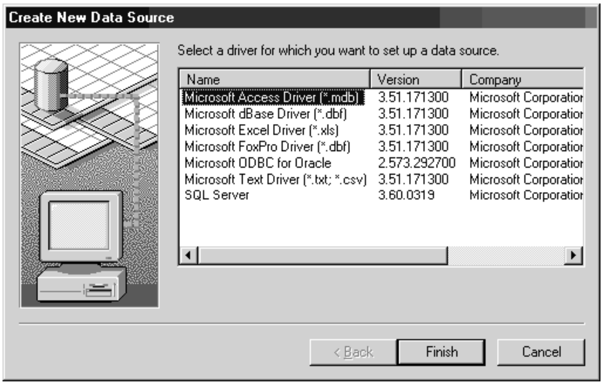
כדי שיתאפשר לך להגדיר מקור נתונים לצורך שימוש בו במסגרת יישום, תידרש לדעת באיזה מנהל התקן יהיה עליך להשתמש ואיזו תצורה יהיה עליך להגדיר עבור מנהל התקן זה. לדוגמה, תידרש לדעת את שם קובץ SQL Server או MDB שבו מאוחסנים הנתונים. תידרש גם להגדיר שם ייחודי שיזהה את מקור הנתונים.

נגדיר מקור נתונים לדוגמה כעת. עבור לכרטיסיה User DSN ולחץ על הלחצן Add, כדי ליצור מקור נתונים חדש. תוצג לפניך תיבת הדו-שיח Create New Data Source המוצגת בתרשים 27.5. בתיבת דו-שיח ראשונה זו תבחר את מנהל התקן ODBC שישמש אותך לצורך גישה לנתונים.

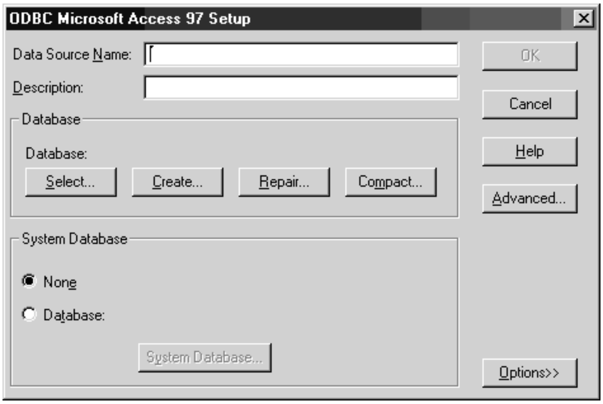
לאחר שתבחר מנהל התקן ותלחץ על הלחצן Finish, תוצג לפניך תיבת דו-שיח כמו זו הנראית בתרשים 27.6 שתאפשר לך להגדיר מסד נתונים מסוג המשמש לעבודה עם מנהל ההתקן שבחרת בתיבת הדו-שיח הקודמת.

בתיבת דו-שיח זו הזן שם בתיבת הטקסט Data Source Name. השם שתזין ישמש את היישום שלך לצורך פנייה לאותו מקור נתונים. תוכל גם להזין Description (תיאור) עבור מקור הנתונים. לסיום לחץ OK.

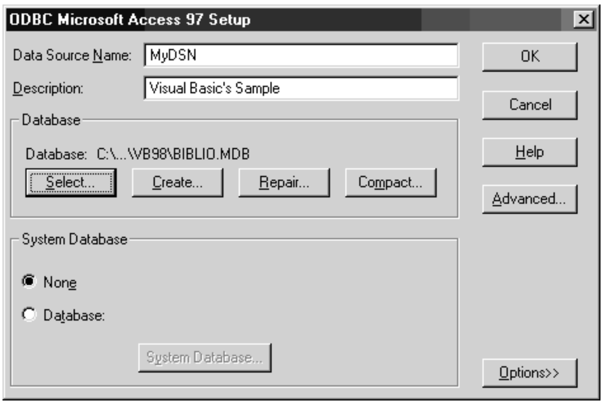
לאחר שתגדיר את השם, תידרש להגדיר באיזה קובץ מסד נתונים או באיזה שרת מסד נתונים תרצה להשתמש בשילוב עם התוכנית שלך. כדי לבחור Access Driver, לחץ על הלחצן Select שבתיבת הדו-שיח. תוצג לפניך תיבת הדו-שיח Select Database (שהיא למעשה תיבת דו-שיח לפתיחת קובץ). נסה את התיבה על ידי בחירת קובץ MDB המאוחסן במחשב שלך. תרשים 27.7 מתאר מקור נתונים בשם MyDSN כשהוא מקושר למסד הנתונים BIBLIO שמסופק יחד עם Visual Basic.



תרשים 27.5: בחירת מנהל התקן ODBC היא הצעד הראשון בהגדרת מקור נתונים

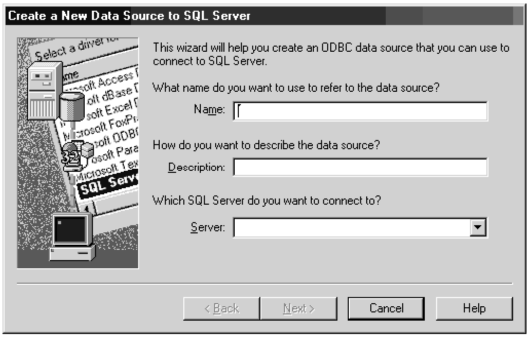


תרשים 27.6: תיבת הדו-שיח Setup תאפשר להגדיר נתונים הדרושים לשם התקשרות למקור נתונים ODBC



תרשים 27.7: תיבת הדו-שיח ODBC Microsoft Access 97 Setup תאפשר לך לבחור את קובץ MDB שאיתו תעבוד

זכור שתיבות הדו-שיח המשמשות להגדרת מקורות נתונים תלויות בסוג מנהל ההתקן שבחרת. אך עדיין תידרש להגדיר שם למקור הנתונים ואת מקום אחסון הנתונים. תרשים 27.8 יציג את תיבת הדו-שיח המשמשת לעבודה עם Microsoft SQL Server.



תרשים 27.8: תיבת הדו-שיח Microsoft SQL Server דורשת לציין שרת

הכרטיסיות להגדרת DSN מאפשרות גם להגדיר תצורות עבור מקורות נתונים של ODBC ולהסיר מקורות נתונים שהתקנת. כדי לשנות הגדרת מקור נתונים, בחר אותו ולחץ על הלחצן Configure. תוצג לפניך אותה תיבת דו-שיח שבה נעזרת בעת הגדרת מקור הנתונים. למחיקת מקור נתונים, בחר אותו ולחץ על הלחצן Remove.

שימוש באובייקטי DAO להגדרת מקור נתונים

הגדרת מקור נתונים אינה חייבת להתבצע אינטראקטיבית. ייתכנו מצבים, למשל התקנת יישומים, שתרצה להוסיף מקור נתונים תוך שימוש בקוד. תוכל להשתמש בשיטה RegisterDatabase של האובייקט DBEngine. תחביר השיטה הוא:

```
DBEngine.RegisterDataBase dbname, driver, silent, attributes
```

טבלה 27.1: הפרמטרים של שיטה RegisterDatabase

פרמטר	הגדרה
dbName	מחרוזת הניתנת להגדרה על ידי המשתמש, מציינת שם מקור נתונים (לדוגמה, "MyDatabase")
driver	ביטוי מחרוזת המציין את שם מנהל ההתקן המותקן במערכת, כפי שהוא מוצג בכרטיסיה ODBC Drivers (לדוגמה, ORACLE)
silent	ערך True בפרמטר מציין שהפרמטר הבא (attributes) מכיל את כל נתוני הקישור. False מציג את תיבת הדו-שיח Setup של מנהל ההתקן המתאים, ומורה למערכת להתעלם מתכונות הפרמטר attributes
attributes	מכיל את כל נתוני הקישור הדרושים לשם שימוש במנהל התקן ODBC. אם בפרמטר silent מוצב False, המערכת מתעלמת מפרמטר זה

קטע הקוד הבא ידגים שימוש בשיטה RegisterDatabase לצורך קישור למסד נתוני Access. לפני השימוש ב- Data Access Objects זכור להוסיף לפרויקט שלך את ההפניה הדרושה לשם כך.

```
Dim sAttrib As String
        Dim sDriver As String
        sAttrib = "DBQ=D:\VB6\BIBLIO.mdb"
        sDriver = "Microsoft Access Driver (*.mdb)"
 DBEngine.RegisterDatabase "MyDSN", sDriver, True, sAttrib
```

לאחר שתפעיל קטע קוד זה, חזור לתיבת הדו-שיח Data Source Administrator, כדי לוודא שנוסף לו DSN חדש מסוג User.

הערה:

ניתן לבצע רישום Remote Data Objects **גם על ידי שימוש בשיטה** rdoRegisterDataSource **של אובייקט** rdoEngine.

שים לב לכך שבעת שימוש במנהל ההתקן של Access, הפרמטר DBQ מציין את שם קובץ מסד הנתונים. אם תרצה לוודא אילו פרמטרים דרושים למנהל התקן מסוים של ODBC, תוכל להגדיר חיבור תוך שימוש ביישום ODBC Manager ולעיין בהגדרות החיבור המופיעות ברישום המערכת. את ההגדרות תוכל לאתר תחת הפריט HKEY_USERS\Default\Software\ODBC\ODBC.INI (תוכל להשתמש בכלי העזר REGEDIT לעיון ברישום המערכת). בעת הגדרת מספר פרמטרים עבור השיטה RegisterDatabase, הפרד בין הפרמטרים באמצעות פסיקים.

אובייקטי נתונים מרוחקים

Data Access Objects (DAO - אובייקטי גישה לנתונים) הם שכבה הפועלת מעל ODBC API. לפני פיתוח Remote Data Objects (RDO - אובייקטי גישה לנתונים מרוחקים), מפתחי Visual Basic רבים נהגו לדלג על שכבה זו ולבצע קריאות ישירות ל- ODBC API. הסיבה שבעטיה בוצעו הקריאות הישירות היתה כמובן רצון להאיץ את פעולת התוכניות. אך השימוש בקריאות ל-API הוא פחות נוח משימוש ב- Data Access Objects. Remote Data Objects פתרו את בעיית אי הנוחות על ידי כך שהן סיפקו ממשק לפניה ל-API אשר התבסס על פעולות מוכרות כגון הגדרת מאפיינים וקריאה לשיטות. האפשרות להשתמש במאפיינים ובשיטות (אשר משמשים במסגרת כל התוכניות הכתובות ב- Visual Basic), הקלה על מפתחים להבין וליישם מושגים הקשורים בגישה למסדי נתונים מבוססי ODBC.

השוואה בין RDO ו-DAO

Remote Data Objects (RDO) דומים מאוד לאובייקטי Data Access Objects (DAO) שתוארו בפרק 26 **שימוש באובייקטי גישה לנתונים (DAO)**. הדמיון הקיים בין טכנולוגיות אלו לא רק מקל על מפתחים להבין את טכנולוגיית RDO אלא גם מקל מאוד על הסבת תוכניות המסתמכות על מסדי נתונים מבוססי PC לפעולה בסביבת שרת/לקוח. למעשה, לאחר יצירת החיבור למסד הנתונים, אותן שורות קוד ששימשו לגישה לנתונים בעת שימוש DAO, תוכלנה לשמש גם לגישה לנתונים תוך שימוש ב-RDO. לשם המחשת הדמיון הקיים בין הטכנולוגיות, טבלה 27.2 תפרט כמה סוגי אובייקטים המוגדרים במסגרת RDO ואת סוגי האובייקטים המקבילים להם המוגדרים במסגרת DAO.

בנוסף לכך, האובייקט rdoResultset תומך בכמה סוגים של סדרות רשומות מוחזרות, אשר דומים לסוגים של מערכי רשומות שנתמכים על ידי אובייקט Recordset. טבלה 27.3 תפרט את נקודות הדמיון בתחום הזה.

טבלה 27.2: אובייקטים של טכנולוגיית RDO ומקביליהם בטכנולוגיית DAO

אובייקט בטכנולוגיית RDO	אובייקט בטכנולוגיית DAO
rdoEngine	DBEngine
rdoEnvironment	Workspace
rdoConnection	Database
rdoTable	TableDef
rdoResultset	Recordset
rdoColumn	Field
rdoQuery	QueryDef
rdoParameter	Parameter

טבלה 27.3: סוגי אובייקט rdoResultset **וסוגים מקבילים של אובייקט** Recordset

rdoResultset	Recordset	הגדרה
keyset	Dynaset	אוסף רשומות בר-עדכון, שהתנועה בו אינה מוגבלת
Static	Snapshot	סידרה שאינה ניתנת לעדכון רשומות, שהיו קיימות בעת יצירת מערך הרשומות. סדרת רשומות זו לא תשקף פעולות עדכון שתבוצענה על ידי משתמשים אחרים
Dynamic	**אין**	דומה ל-Keyset
Forward-only	Forward-only	דומה לסדרת תוצאות סטטית (Static Resultset) או למערך רשומות מסוג Snapshot שניתן לנוע בו רק קדימה. זהו סוג ברירת מחדל של אובייקט Resultset

שים לב לכך שטכנולוגיית RDO אינה תומכת בסוג כלשהו של אובייקט rdoResultset שמחזיר Table. זאת היות ואובייקטי גישה לנתונים מרוחקים נוהגים להיעזר במשפטי SQL לשם אחזור נתונים מטבלה או מכמה טבלאות. את סדר הרשומות באובייקט rdoResultset יש להגדיר על ידי הכללת קטע Order By במסגרת משפט SQL שישמש ליצירת סדרת הרשומות. כמו כן, היות ולא קיימת במסגרתה מהות המקבילה ל-Table, RDO גם אינו תומך באינדקסים.

כפי שניתן היה לצפות עקב הדמיון באובייקטים, קיימות שיטות המוגדרות לגבי RDO אשר דומות לשיטות המוגדרות לגבי DAO. שיטות אלו והאובייקטים שבמסגרתן הן מוגדרות יפורטו בטבלה 27.4.

טבלה 27.4: שיטות של אובייקט RDO **ושיטות מקבילות של אובייקט** DAO

שיטת RDO	אובייקט RDO	שיטת DAO	אובייקט DAO
rdoCreateEnvironment	rdoEngine	CreateWorkspace	DBEngine
BeginTrans	rdoConnection	BeginTrans	Workspace
CommitTrans	rdoConnection	CommitTrans	Workspace
OpenConnection	rdoEnvironment	OpenDatabase	Workspace
RollbackTrans	rdoConnection	Rollback	Workspace
CreateQuery	rdoConnection	CreateQueryDef	Database
Execute	rdoConnection	Execute	Database
OpenResultset	rdoConnection	OpenRecordset	Database

השיטות שתפורטנה להלן משותפות לאובייקט rdoResultset ולאובייקט Recordset:

- AddNew - מוסיפה שורה (רשומה) חדשה לקבוצת רשומות.
- Delete - גורעת את השורה (הרשומה) הנוכחית מקבוצת רשומות.
- Edit - מכינה את השורה הנוכחית לשינוי בנתונים הב בה.
- MoveFirst - עוברת לשורה הראשונה בקבוצת רשומות.
- MoveLast - עוברת לשורה האחרונה בקבוצת רשומות.
- MoveNext - עוברת לשורה הבאה בקבוצת רשומות.
- MovePrevious - עוברת לשורה הקודמת בקבוצת רשומות.
- Update - מעבירה את השינויים שבוצעו ברשומה בתוך מאגר ההעתקה לרשומה עצמה. מאגר ההעתקה הוא מקום בזיכרון שבו שמורים הערכים המאוחסנים ברשומה שעמה עובדים כעת.

גישה למסד נתונים תוך שימוש ב-RDO

לשם המחשה נוספת של הדמיון הרב הקיים בין RDO ל-DAO נשתמש בקטעי הקוד שיובאו בתוכנית 27.1 ובתוכנית 27.2 לשם ביצוע אותה פעולה על-מסד הנתונים BIBLIO. ההבדל בין שני קטעי הקוד הוא באובייקטים ובשיטות שישמשו ליצירת הרשומות שתוחזרנה במסגרתם. לאחר יצירת אובייקט Recordset או אובייקט Resultset, השורות הנותרות בקטע הקוד תשמשנה להדפסת ערך בשדה הראשון כל רשומה. את מקור הנתונים, MyDSN, שאליו מתייחסת דוגמת השימוש ב-RDO, יצרנו קודם לכן, בעזרת ODBC Manager.

הערה:

כדי שיתאפשר לך להשתמש באובייקטי גישה לנתונים מרוחקים, תידרש להוסיף לפרויקט שלך הפניה ל- Microsoft Remote Data Object, על ידי שימוש בתפריט Project References.

תוכנית 27.1: RDOSAMPLE.TXT - גישה לנתונים שבמקור נתוני ODBC תוך שימוש בשיטות RDO

```
Dim db As rdoConnection
Dim rs As rdoResultset
Dim sSQL As String

Set db = rdoEngine.rdoEnvironments(0).OpenConnection("MyDSN")

sSQL = "Select * From Titles"
Set rs = db.OpenResultset(sSQL, rdOpenKeyset)

rs.MoveFirst
```

```
Do While Not rs.EOF
        Print rs.rdoColumns(0)
rs.MoveNext
loop

rs.Close
db.Close
```

תוכנית 27.2: DAOSAMPLE.TXT - גישה לנתונים שבמקור נתוני ODBC תוך שימוש בשיטות DAO

```
Dim db As Database
Dim rs As Recordset
Dim sSQL As String

Set db = DBEngine.Workspaces(0). OpenDatabase("BIBLIO.MDB")

sSQL = "Select * From Titles"
Set rs = db.OpenRecordset(sSQL, dbOpenDynaset)

rs.MoveFirst
Do While Not rs.EOF
        Print rs.Fields(0)
        rs.MoveNext
loop

rs.Close
db.Close
```

ביצוע פעולות במסגרת מסד הנתונים באופן אסינכרוני הוא נושא נוסף שייתכן שתרצה ללמוד באמצעות שימוש ב-RDO. במהלך ביצוע פעולות באופן אסינכרוני, השליטה על המערכת מוחזרת ליישום שלך לפני השלמת הפעולה שבמסגרת מסד הנתונים, כמו בדוגמה הבאה:

```
Set db = rdoEngine.rdoEnvironments(0).[ic:ccc]
OpenConnection("MyDSN",,,, rdAsyncEnable)
While db.StillConnecting = True
        Print "Connecting..."
Wend
```

הקבוע rdAsyncEnable מגדיר פעולה אסינכרונית. לולאת Do While תמשיך בפעולתה עד שהחיבור יווצר ובמאפיין StillConnecting יוצב ערך False.

שימוש בפקד RemoteData

אם תרצה אמצעי מהיר יותר ליצירת יישומים, תוך שימוש במקור נתונים של ODBC, תוכל להשתמש בפקד Remote Data Control (RDC - פקד הגישה לנתונים מרוחקים). פקד RDC דורש ממך להגדיר עבורו כמה מאפיינים ולאחר מכן הוא מטפל עבורך בכל מה שנדרש לשם יצירת חיבורים למקורות נתונים של ODBC. כך הופך הפקד את ביצוע השיטות על אובייקטי RDO לאוטומטי באותו אופן בו פקד Data הופך את הביצוע השיטות לגבי Data Access Objects לאוטומטי.

לאחר שתגדיר את פקד Remote Data Control תוכל להיעזר בפקדי איגוד להצגת הנתונים שייכללו ב-Resultset שיווצר על ידי פקד Data. את הפקדים המאוגדים תגדיר כשם שהגדרת אותם לעבודה עם פקד הנתונים, ראה פרק 25 **פקד הנתונים ופקדי איגוד נתונים**, פרט לכך שכעת המאפיין DataSource של הפקד המאוגד יצביע על פקד מסוג RemoteData. לאחר שהפקדים המאוגדים יוגדרו, הם יתעדכנו לאחר כל גישה לשורה בקבוצת הרשומות, שתבוצע על ידי פקד RemoteData.

השוואה בין RDC לפקד הנתונים

באחד הסעיפים הקודמים השוונו בין Remote Data Objects לבין Data Access Objects. כעת נתייחס לנקודות הדמיון הקיימות בין פקד הנתונים לבין Remote Data Control. כפי שניתן לצפות, למאפיינים רבים של Remote Data Control יש מקבילים בקרב המאפיינים המוגדרים עבור פקד הנתונים. מאפיינים אלה ואופני פעולתם יפורטו בטבלה 27.5.

טבלה 27.5: השוואה בין מאפייני פקד Remote Data לבין מאפייני פקד Data

מאפיין RDC	מאפיין פקד Data	מטרה
BOFAction	BOFAction	בודק האם הדגל שמציין את תחילת הקובץ מסומן עקב הפעלת השיטה MovePrevious, כאשר מצביע הרשומות מצביע על הרשומה הראשונה בקובץ
DataSourceName	DatabaseName	מציין באיזה מסד נתונים כלולים הנתונים
EOFAction	EOFAction	בודק האם הדגל המציין את סוף הקובץ מסומן עקב הפעלת השיטה MoveNext, כאשר מצביע הרשומות מצביע על הרשומה האחרונה בקובץ
ResultType	RecordsetType	מגדיר את סוג קבוצת הנתונים (Dataset) שהפקד ייצור
SQL	RecordSource	מכיל את משפט SQL אשר מגדיר את הנתונים שאותם יש לאחזר

הגדרת פקד RDC

הגדרת פקד RDC דומה מאוד להגדרת פקד הנתונים. לפני שיתאפשר לך להשתמש בפקד RDC תידרש להוסיף את הפקד לפרויקט שלך. עשה זאת בעזרת תיבת הדו-שיח Components, שאותה תציג על ידי בחירה בפקודה **Components** מתפריט **Project**. לאחר שתסגור את תיבת הדו-שיח, הפקד יתווסף לארגז הכלים שלך.

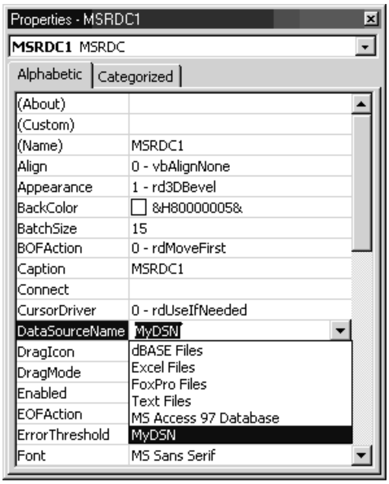
הערה:

פקד RDC זמין לבחירה רק במסגרת גרסת Visual Basic Enterprise. **ייתכן** גם שבחרת לבצע התקנה מותאמת אישית של גרסת Enterprise, שבמסגרתה בחרת שלא להתקין את פקד RemoteData. אם נהגת כך, הפקד Remote Data **לא יוצג** בתיבת **הדו-שיח** Components **ותידרש להתקין** שוב קטע זה.

בצע את הפעולות הבאות כדי להגדיר את פקד Remote Data:

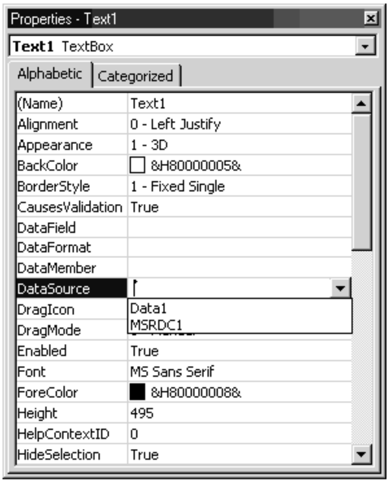
1. שרטט פקד RemoteData על גבי טופס.
2. הצב במאפיינים Name ו-Caption של הפקד ערכים שיהיו בעלי משמעות עבורך.
3. הגדר את המאפיין DataSourceName של הפקד. את ערך המאפיין תוכל להזין באמצעות הקלדה או על ידי בחירתו מהרשימה הנפתחת.
4. הצב במאפיין SQL של הפקד משפט SQL חוקי, אשר יגדיר את הנתונים הדרושים לך.

בסעיף 3, ציינו שתוכל לבחור ערך עבור המאפיין DataSource מתוך רשימה נפתחת. הרשימה הנפתחת שתוצג לפניך תכיל את שמות כל מקורות הנתונים של ODBC שהוגדרו במערכת שלך. בתרשים 27.9 תוכל לראות דוגמת רשומה כזו.



תרשים 27.9: בעת הגדרת מאפיין DataSourceName, **יתאפשר** לך לבחור ערך עבור המאפיין מתוך רשימת מקורות הנתונים המותקנים במערכת

לאחר שתגדיר פקד RemoteData תוכל לקשר אליו פקדי איגוד נתונים, על ידי הגדרת המאפיין DataSource של הפקדים. בתרשים 27.10 תוכל לראות שהרשימה הנפתחת בשורת הגדרת המאפיין, מכילה את שמות כל פקדי Remote Data ו-Data שבטופס הנוכחי. לאחר שתגדיר את מאפיין DataSource, תוכל לבחור את הגדרת המאפיין DataField מתוך רשימה, כפי שעשית בעת קישור הפקדים לפקד Data.



תרשים 27.10: הרשימה הנפתחת בשורת המאפיין DataSource מכילה את כל פקדי הגישה לנתונים, מסוג Data ומסוג Remote Data, הזמינים לבחירה

מכאן...

פרק זה הקנה לך הבנה בסיסית של המושג **יישומי שרת/לקוח** (Client/Server Applications). הפרק גם המחיש לך כיצד אובייקטי Remote Data ופקדי Remote Data יכולים להקל עליך את הגישה למסדי נתונים המבוססים על ODBC. מסדי נתונים כאלה כלולים בתוכניות רבות הפועלות בשיטת שרת/לקוח. מידע נוסף אודות נושאים שנדונו בפרק זה תוכל למצוא בפרקים הבאים:

- מידע אודות מושגי יסוד בתחום מסדי הנתונים תוכל למצוא בפרק 24 **יסודות מסד הנתונים**.
- מידע אודות כלי הגישה לנתונים שיחליפו בעתיד את טכנולוגיות DAO ו-RDO תוכל למצוא בפרק 28 **גישה לאובייקטי נתונים באמצעות פקדי ActiveX**.
- מידע נוסף אודות שפת SQL תוכל למצוא בנספח 3 **תקציר פקודות SQL**.

28

גישה לאובייקטי נתונים באמצעות פקדי ActiveX (ADO)

מה בפרק?

- **מבוא ל-ADO**
- **שימוש ב- ADO Data Control**
- **שימוש בפקד DataGrid**
- **שימוש ב- ActiveX Data Objects**
- **מערכי רשומות מנותקים**

גירסה 2.0 של טכנולוגיית ActiveX Data Objects - **ADO** היא התוספת החדשה ביותר למערך כלי הגישה לנתונים של Visual Basic. טכנולוגיית ADO מאפשרת לך לקיים אינטראקציה עם מסדי נתונים מסוגים שונים, תוך שימוש בתוכניות הכתובות ב-Visual Basic, ברכיבי ActiveX או בעזרת Active Server Pages. פרק זה יציג בפניך את טכנולוגיית ADO, וידגים את היכולות החשובות ביותר של טכנולוגיה זו.

מבוא ל-ADO

טכנולוגיית הגישה לאובייקטי נתונים באמצעות פקדי ActiveX (ActiveX Data Objects) המכונה בקיצור ADO היא התוספת החדישה ביותר למאגר טכנולוגיות הגישה לנתונים של Visual Basic. טכנולוגיית ADO נועדה להחליף את טכנולוגיית Data Access Objects (DAO), שהיתה הטכנולוגיה הראשונה לגישה לנתונים שיושמה במסגרת Visual Basic ואת טכנולוגיית Remote Data Access (RDO) שהיא חלופה מהירה יותר לטכנולוגיית DAO. בדומה לשתי הטכנולוגיות אלו, גם ADO מציעה מספר אפשרויות לגישה לנתונים. בפרק זה נסקור את הפעולות ואת השיטות הדרושות לשם התקשרות לנתונים תוך שימוש בטכנולוגיית ADO.

שיטות התקשרות לנתונים

ADO היא אמצעי המאפשר לקוד שתיצור, לגשת למסדי נתונים. אך כיצד היא עצמה מתקשרת למסדי נתונים? התשובה היא שימוש בגישת **OLE DB Provider**. OLE DB הוא ממשק מסד הנתונים היסודי החדש של Microsoft, אשר מאפשר גישה לסוגים שונים של נתונים. כיום מוצעים OLE DB Providers עבור מסדי נתונים רגילים (כגון SQL Server) ועבור מקורות אחרים של נתונים, כגון שרתי דואר אלקטרוני. אובייקט OLE DB Provider משמש כדי לחשוף את מסדי הנתונים האלה בפני טכנולוגיית ADO, המאפשרת לך להתקשר לנתונים באופנים הבאים:

- **פקד ADO**. פקד ADO הוא פקד ייעודי אשר מטפל בתקשורת עם מסד נתונים. כל מה שתידרש לעשות לשם פנייה למסד הנתונים הוא להגדיר כמה ממאפייניו ולקשר אליו פקדים איגוד נתונים שישמשו להצגת הנתונים.
- **ממשק אובייקטים** כאשר אתה מוסיף לפרויקט הפניה ל-ADO, סדרת אובייקטים שלמה הופכת זמינה לשימוש התוכנית שלך. תוכל לבצע פעולות על הנתונים ישירות מקוד, או לבצע פעולות משולבות של פקדים ואובייקטים.

הגדרת פקד **ADO** היא פשוטה ומהירה, אך השימוש בממשק האובייקטים מקנה לך עוצמה וגמישות רבה יותר. בפרק זה נבחן את שני האופנים המשמשים לשם פנייה לטכנולוגיית ADO.

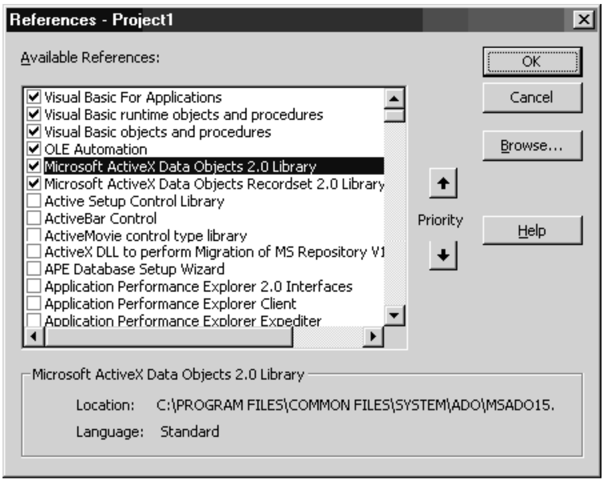
התקנה

על סמך ניסיון העבר אני יכול להעריך שעד שספר זה יגיע לידיך, Microsoft כבר תשחרר תיקונים (Patches) וחבילות שירות (Service Packs) עבור רבים ממוצריה. כך שלפני שתתחיל להשתמש בטכנולוגיית ADO, כדאי שתוריד מה-Web עותק של הגירסה העדכנית ביותר של ספריה זו, ותתקין אותו במחשב שלך. נכון לכתיבת שורות אלה, Visual Basic 6.0 כוללת את ADO 2.0 ואילו משתמשי Visual Basic 5 יוכלו להוריד מה-Web את גירסה 1.5c. חבילת Microsoft Data Access Components (MDAC - רכיבי גישה לנתונים של Microsoft) כוללת קבצים, תיעוד של ADO, גירסה העדכנית ביותר של ODBC, ואת ספריית Remote Data Services. את החבילה תוכל למצוא באתר: http://www.microsoft.com/data/ado.

טיפ:

בעת ביצוע פעולות התקנה של גרסאות עדכון לכלי הגישה לנתונים, בחר את האפשרות Custom Installation (התקנה מותאמת אישית), והקפד להתקין גם את קבצי התיעוד של ADO ואת Access Database Driver.

בתיבת הדו-שיח References כלולים שני פריטים שיש לבחור אותם בעת עבודה עם ADO: Microsoft ActiveX Data Objects 2.0 Library ו- Microsoft ActiveX Data Recordset 2.0 Library. כדי שיתאפשר לך להפעיל את הדוגמאות שתובאנה במהלך הפרק, הקפד להוסיף לפרויקטים שלך את ההפניות הדרושות, כמתואר בתרשים 28.1.



תרשים 28.1: להוספת הפניה ל-ADO, בחר References מתפריט Project

ההפניה ל- Microsoft ActiveX Data Objects 2.0 שבתיבת הדו-שיח References מכילה את כל מיגוון האובייקטים בעלי היכולות המלאות של ADO. ההפניה השנייה שאותה תבחר מכילה רק קישורים לאובייקטים מסוג Recordset, המיועדים לשימוש עם Remote Disconnected Recordsets (מערכי רשומות מנותקים).

הגדרת מקור נתונים

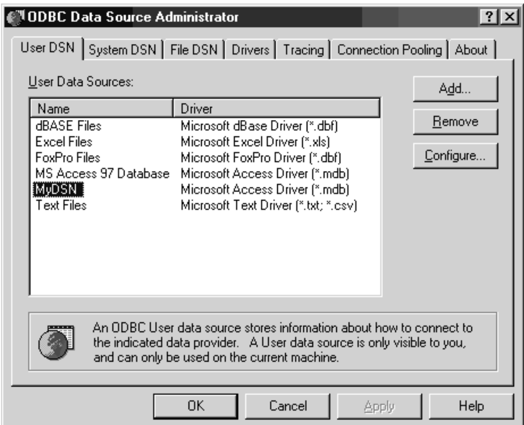
בכל מצב בו יהיה עליך להשתמש בנתונים במסגרת תוכנית, הנתונים האלה יבואו ממקור **נתונים** (Data Source). מקור נתונים יכול להיות כל מסד נתונים, החל ממסד נתוני Access קטן וכלה במסד נתוני AS/400. כיום מוצעים מוצרים רבים לעבודה עם מסדי נתונים, שכמעט לכולם תוכל להתקשר מתוך Visual Basic. ODBC הוא אחד האמצעים המאפשרים לך להתקשר למסדי נתונים מסוגים שונים. תוכל להגדיר מנהל התקן (Driver) של ODBC במערכת Windows, להשתמש בו לצורך גישה למסד הנתונים מתוך Visual Basic.

כדי שיתאפשר לך לעבוד עם הדוגמאות שתובאנה בפרק זה, תידרש להגדיר מקור נתונים עבור מסד הנתונים BIBLIO.MDB שבחבילת Visual Basic. התחל את פעולת ההגדרה על ידי לחיצה על הסמל המסומן בתווית 32-bit ODBC שבלוח הבקרה של Windows. תוצג לפניך תיבת הדו-שיח ODBC Data Source Administrator המוצגת בתרשים 28.2.

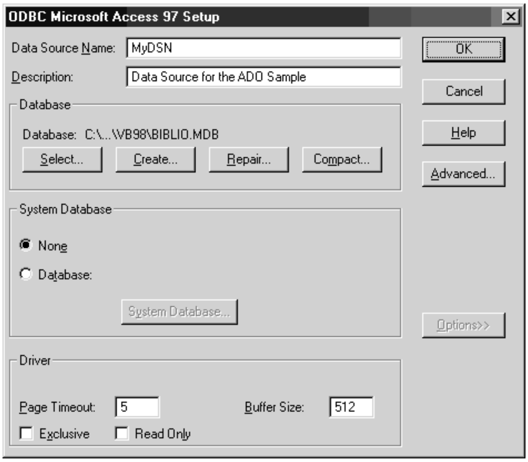
הערה:

תוכל גם להגדיר חיבורים מבלי להגדיר DSN. מידע אודות אופן פעולה זה תוכל למצוא בסעיף "הגדרת חיבורים באמצעות ADO", שיובא בהמשך הפרק.

ליצירת מקור נתונים חדש, לחץ על הלחצן Add. מסד הנתונים BIBLIO.MDB הוא מסד נתוני Access, כך שיהיה עליך לבחור את מנהל ההתקן של Access וללחוץ על לחצן Finish. כעת תוצג תיבת דו-שיח של מנהל התקן ODBC של Access (Access ODBC Driver). לחץ על לחצן Select וחפש את מסד הנתונים BIBLIO.MDB (ברוב המקרים מסד הנתונים הזה יהיה מאוחסן בתיקיית היישומים של Visual Basic). הגדר את שם מקור הנתונים BIBLIO ואם תרצה תוכל גם להזין תיאור של מסד הנתונים. לאחר שתסיים להזין את הנתונים, מסך ההגדרה ייראה כמו בתרשים 28.3.



תרשים 28.2: ODBC Data Source Administrator מאפשר להגדיר Data Source Names שיצביעו על מסדי נתונים ספציפיים



תרשים 28.3: התחברות למסד נתוני Access באמצעות ODBC שונה מעט מהתחברות באמצעות Jet. ההבדלים פורטו בפרק 26, **"שימוש באובייקטי גישה לנתונים (DAO)"**

לחץ על OK כדי לסגור את ODBC Data Source Administrator. כעת, לאחר שהגדרת את תצורת מקור הנתונים, תוכל להשתמש בנתונים המאוחסנים במקור הנתונים BIBLIO, על ידי שימוש באובייקטי ADO.

הערה:

ODBC מהווה אומנם אמצעי נוח להגדרת מקורות נתונים, אך ניתן להשתמש בטכנולוגיית ADO גם מבלי להשתמש ב-ODBC. כיום מוצע אובייקט מסוג ADO Data Provider המיועד לשימוש יחד עם מסדי נתונים ללא שימוש ב-ODBC.

שימוש בפקד ADO Data

ל- Visual Basic 6.0 סוג חדש של פקד נתונים, ADO Data Control. פקד זה ממלא אותו תפקיד שממלא פקד Data, אך הוא עושה זאת תוך שימוש בטכנולוגיית ADO.

קודם לכן כבר ציינו כי ADO ניתנת להפעלה הן על ידי שימוש בקוד, הן על ידי שימוש בפקדים, והן תוך שילוב שני האופנים. פקד Data אומנם יכול לשמש כפקד עצמאי, אך אופן השימוש השכיח ביותר בפקדים כאלה הוא לשם אספקת נתונים לפקדים אחרים שבטופס. ניתן לקחת פקד רגיל כלשהו, לדוגמה פקד תיבת טקסט ו"לאגד" אותו לפקד Data. לאחר פעולה כזו, פקד תיבת הטקסט מכונה **פקד איגוד** או **פקד איגוד נתונים** (Bound Control). הפקדים המאוגדים מקושרים למסד הנתונים ותוכנם משקף את תוכן הרשומה ההנוכחית במסד הנתונים, המועבר אליהם באמצעות פקד Data.

הערה:

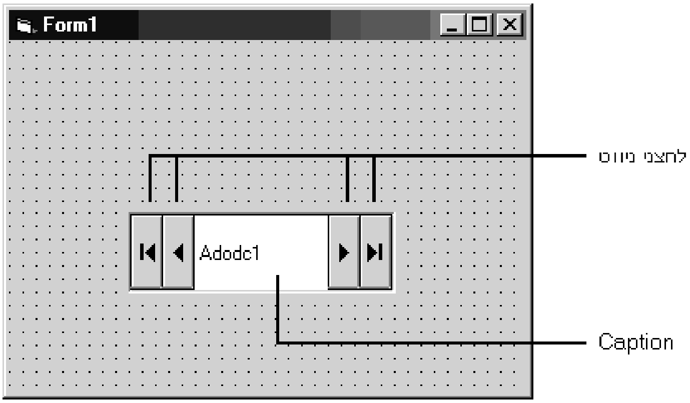
אם לא קראת את פרק 25 **פקד הנתונים ופקדי איגוד נתונים**, ייתכן שתרצה לקרוא אותו כעת. פרק זה דן בפקד הנתונים הרגיל ובפקדי איגוד נתונים.

הגדרת פקד ADO Data

הדרך הפשוטה ביותר ללמוד כיצד להשתמש בפקד ADO היא על ידי הפעלתו בפרויקט דוגמה. בסעיף זה תיצור פרויקט דוגמה שילמד אותך אודות ADO Data Control. התחל את הפעולה על ידי כך שתפעיל את Visual Basic, ותיצור פרויקט חדש מסוג Standard .EXE

כדי שיתאפשר לך להשתמש בפקד ADO Data, יהיה עליך להוסיף את הפקד לארגז הכלים של Visual Basic. הוסף את הפקד לארגז הכלים על ידי לחיצה ימנית על שטח ריק בארגז הכלים, בחירה בפקודה **Components** מהתפריט תלוי ההקשר שיוצג וסימון ליד הפריט ADO Data Control 6.0 בתיבת הדו-שיח שתופיע.

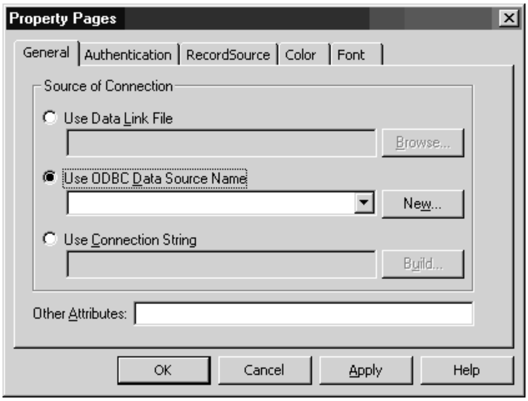
לאחר מכן, שרטט מופע של הפקד ADO Data על הטופס. פקד ADO Data, אשר מוצג בתרשים 28.4 דומה מאוד לפקד הנתונים הרגיל.



תרשים 28.4: הפקד ADO Data מקשר את התוכנית שלך למקור נתונים של ADO

לפני שתמשיך בהגדרת הפקד, כדאי שתתייחס לרגע למראה ההתחלתי שלו. ארבעת לחצני החיצים, הדומים ללחצנים שבמכשירים להשמעת תקליטורים, משמשים להחלפת הרשומה הנוכחית של מסד הנתונים. כותרת הפקד, שניתן להגדיר באמצעות המאפיין Caption, משמשת להעברת מידע למשתמש ולכן כדאי להציב בה כותרת בעלת-משמעות. הכותרת ההתחלתית שתוצג על הפקד, ADODC1 היא שם ברירת המחדל של פקד ADO Data הראשון שהוסף לטופס.

כדי שיתאפשר לך להתקשר למסד הנתונים ולאחזר ממנו נתונים, תידרש להגדיר כמה ממאפייני הפקד. את רוב המאפיינים האלה תוכל להגדיר באמצעות תיבת הדו-שיח Property Pages (גיליונות מאפיינים) של הפקד, אשר מוצגת בתרשים 28.5. הצג את תיבת הדו-שיח על ידי לחיצה ימנית על הפקד ובחירה בפקודה ADODC Properties מהתפריט תלוי ההקשר שיוצג.



תרשים 28.5: היעזר בגיליון המאפיינים של ADO Data Control, כדי להגדיר את הפקד במהלך עיצוב היישום

תיבת הדו-שיח גיליונות המאפיינים של פקד ADO Data מכילה את הכרטיסיות המפורטות להלן:

- **General** (כללי). מגדירה את אופן החיבור שיוגדר בין הפקד למסד הנתונים.
- **Authentication** (אימות זהות). מאפשרת להגדיר שם משתמש וסיסמה שהפקד יוכל להשתמש בהם לשם התקשרות למסד הנתונים, אם דרושים לשם כך שם משתמש וסיסמה.
- **Recordsource** (מקור רשומות). משמשת להגדרת מערך רשומות שהפקד יאחזר ממקור הנתונים. במאפיין זה ניתן להציב שם טבלה, שם שיגרה מאוחסנת או שם שאילתת SQL.
- **Color and Font** (צבע וגופן). שינוי חזות הפקד.

את רוב האופציות שבגיליון המאפיינים ניתן להגדיר גם באמצעות קוד. על ידי הגדרת המאפיינים הרלוונטיים ניתן לקשר פקד ADO Data למסד הנתונים שלך ולהשתמש בפקדי איגוד נתונים שיוספו לטופס, לצורך הצגת הנתונים.

חיבור פקד ADO Data למקור נתונים

כבר ידוע לך שמסדי נתונים מורכבים מטבלאות רבות ויכולים להכיל כמויות נתונים גדולות. לדוגמה, מסד הנתונים BIBLIO כולל טבלה המכילה נתוני סופרים, וטבלה אחרת מכילה רשימת ספרים שנכתבו על ידי אותם סופרים. ברוב המקרים בהם אתה חושב אודות שימוש ב- Visual Basic לעבודה עם מסד נתונים, אינך מתכנן להשתמש בכל הרשומות שבמסד הנתונים, אלא רק בתת-קבוצה של אוסף הרשומות, שבה כלול המידע הרצוי לך.

תת-קבוצה זו (המכונה גם מערך רשומות - Recordset) מוגדרת על סמך ***מקור רשומות*** (Record Source), הבנוי מסדרת קריטריונים המוגדרים על-ידך. לדוגמה, נניח שאתה מעוניין להפעיל על מסד הנתונים שאילתה שתאתר את כל הסופרים ששם המשפחה

שלהם הוא Smith. בדוגמה זו מקור הנתונים הוא מסד הנתונים BIBLIO ומקור הרשומות הוא השאילתה שתחזיר את קבוצת הרשומות שבה יכללו כל המחברים ששם המשפחה שלהם הוא Smith.

סעיף זה נועד כדי להראות לך שפקד ADO Data מציג לשאר התוכנית קבוצת רשומות שנוצרת על ידי הפעלת שאילתה על מסד הנתונים.

יצירת החיבור

כדי שיתאפשר לפקד ADO Data לבצע גישה לנתונים, תידרש קודם לקשר אותו למסד נתונים. עליך להציב במאפיין ConnectionString של הפקד את הנתונים הדרושים לשם כך. את המאפיין הזה תוכל להגדיר הן בזמן עיצוב היישום והן בזמן ריצה. הכרטיסיה General שבגיליונות המאפיינים של הפקד, שהוצגו בתרשים 28.5, תציע לך שלוש דרכים להגדיר את המאפיין Connection String :

- **Use Data Link File** (שימוש בקובץ נתוני קישור). אופציה זו מאפשרת לטעון נתוני קישור שנשמרו בקובץ UDL (Microsoft Data Link).
- **Use ODBC Source Name** (שימוש בשם מקור נתונים של ODBC). אם תבחר באופציה זו, תוכל לבחור את אחד מ-DSN המוגדרים במחשב. תוכל גם ללחוץ על לחצן New כדי להגדיר DSN באותו אופן בו היית עושה זאת דרך לוח הבקרה.
- **Use Connection String** (שימוש במחרוזת חיבור). האופציה הזו תאפשר לך להגדיר את מחרוזת החיבור באופן ישיר. לחיצה על לחצן **Build** תעביר אותך לתיבת דו-שיח שתסייע לך בבניית המחרוזת.

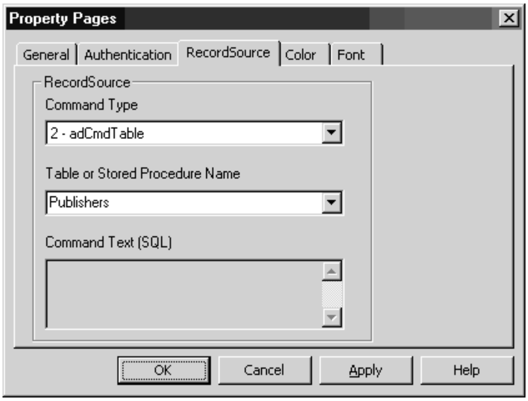
היות שכבר הגדרת את מקור הנתונים המבוסס על מסד הנתונים BIBLIO, בחר את האופציה Use ODBC Source Name ובחר את מסד הנתונים BIBLIO מתוך הרשימה הנפתחת שתוצג.

הגדרת מקור הרשומות

כעת, לאחר שהורית לפקד הנתונים שעליו להתקשר למסד הנתונים BIBLIO, יהיה עליך להורות לו אילו נתונים הוא יידרש להחזיר מתוך מסד הנתונים. בעת עיצוב היישום תוכל לעשות זאת על ידי בחירת הכרטיסיה RecordSource שבתיבת הדו-שיח Property Pages של הפקד, אשר מוצגת בתרשים 28.6.

פקד ADO Data מתנהג כמו אובייקט Command של ADO, שאודותיו תלמד במסגרת הסעיף **אובייקט Command**. קיימים ארבעה סוגי פקודות שבהם תוכל להיעזר כדי לאחזר נתונים :

- adCmdText - מבצעת שאילתת SQL על מקור נתונים.
- adCmdStoredProc - קוראת לשיגרה שמאוחסנת בשרת מסד הנתונים.
- adCmdTable - מגדירה שם טבלה, משמשת כדי להחזיר טבלה שלמה.
- adCmdUnknown - פקודה מסוג בלתי ידוע.



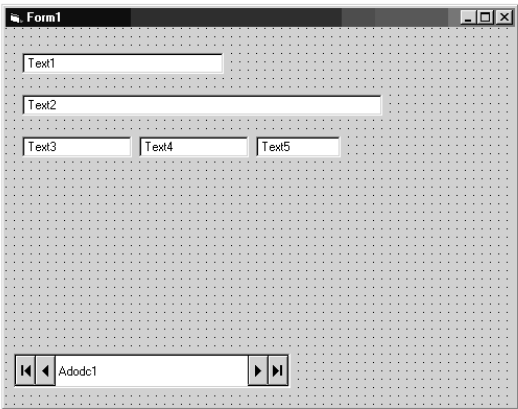
תרשים 28.6: המאפיין RecordSource מורה לפקד ADO Data אילו נתונים לאחזר

תיבת הטקסט התואמת לסוג הפקודה שאותה תבחר תופעל, וכך יתאפשר לך להזין פקודה. לצורך יצירת יישום לדוגמה שיפנה למסד הנתונים BIBLIO, בחר את הטבלה Publishers, בשלמותה.

כדי לבחור את הטבלה, הצב את הערך adCmdTable במאפיין CommandType, בכך שתבחר בו מתוך הרשימה הנפתחת. הבחירה בערך זה תאפשר את הפעלת תיבת הרשימה Table or Stored Procedure Name. בחר מתוך רשימה זו את הטבלה Publishers, כך שתיבת הדו-שיח תיראה כעת כמו תיבת הדו-שיח המוצגת בתרשים 28.6. לבסוף לחץ על OK כדי לסגור את תיבת הדו-שיח ולשמור את השינויים בפקד.

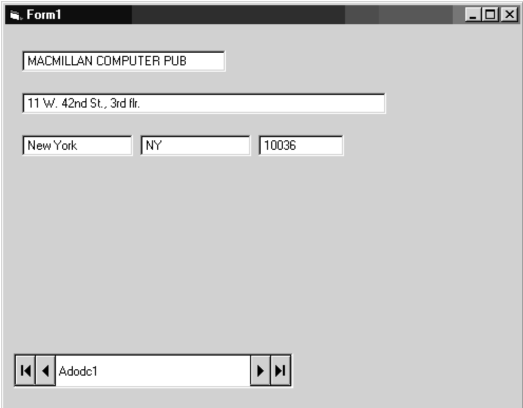
הצגת נתונים

עד כה הגדרת את כל המאפיינים שיש להגדיר כדי שיתאפשר לפקד ADO לייבא נתונים לתוכנית. אך סביר להניח שתרצה גם להציג את הנתונים באופן כלשהו, כדי לאפשר למשתמשים לעיין בהם, לדפדף ברשומות ואולי אפילו לערוך אותן. בפרק 25 **פקד הנתונים ופקדי איגוד נתונים** למדת להשתמש בפקדי איגוד נתונים. לצורך יישום הדוגמה, שרטט כעת חמש תיבות טקסט על הטופס, כמתואר בתרשים 28.7. תיבות טקסט אלו תשמשנה להצגת שם המוציא לאור, הכתובת, העיר, המדינה והמיקוד.



תרשים 28.7: ניתן להצמיד תיבות טקסט לשדות ספציפיים במערך רשומות

כעת יהיה עליך להצמיד כל אחד מתיבות הטקסט לשדה במערך הרשומות. כדי לעשות זאת, הצב את שם פקד ADO Data (ADODC1) במאפיין DataSource של כל אחד מהפקדים, ואת שם השדה המתאים במאפיין DataField של הפקד. לאחר שתסיים, הפעל את יישום הדוגמה. יתאפשר לך לדפדף ברשומות שבמסד הנתונים, להציג ולערוך את הנתונים שבהן, כמתואר בתרשים 28.8.



תרשים 28.8: חזות יישום הדוגמה היא אומנם גולמית מעט, אך זהו יישום יעיל, שלא נדרשת לכתוב קוד כלשהו במסגרת פיתוחו

כאשר תנווט במסד הרשומות, תוך שימוש בלחצני הפקד, תוכן תיבות הטקסט המאוגדות יתעדכן כדי להציג את תוכן הרשימה הנוכחית של מסד הנתונים. אם תרצה להיעזר בפקדי איגוד נתונים לצורך הצגת מספר רשומות ברגע נתון, תידרש להשתמש בפקד ייעודי כדוגמת הפקד DataGrid, שיתואר בסעיף **שימוש בפקד DataGrid**.

שינוי מקור הרשומות באמצעות קוד

זה עתה ראית שקל מאוד להגדיר פקד נתונים בשלב עיצוב היישום. אך גם בפרויקטים הפשוטים ביותר סביר להניח שתידרש להשתמש בפקד באופן יותר דינמי. למרבה המזל תוכל להגדיר את הפקד על ידי ביצוע הפעולות המפורטות להלן:

1. הגדר את המאפיין ConnectionString.
2. הגדר את המאפיינים CommandType ו-RecordSource.
3. הפעל את שיטת Refresh של הפקד כדי לאחזר את הנתונים.

במצבו הנוכחי, פקד ADO Data יוכל לאחזר כל רשומה בטבלה Publishers. אך תוכל גם להוסיף קוד שישנה את הגדרת המאפיין RecordSource כדי לאפשר לפקד להציג מערך רשומות אחר.

נניח שתרצה להציג רק את נתוני המוציאים לאור שכתובותיהם מצויות במדינה מסוימת. תוכל לכתוב שאילתת SQL פשוטה שתציג כתובות אלו, כמו בדוגמה הבאה:

```
Select * from Publishers Where State='NY'
```

במקום שתידרש להגדיר את השאילתה במהלך עיצוב היישום, תוכל להוסיף לפרויקט הדוגמה קוד שיאפשר להגדיר אותה במהלך פעולת התוכנית. ראשית, הוסף לטופס לחצן פקודה. שנה את כותרתו ל- State Lookup ואת שמו ל-cmdState. הוסף את הקוד המובא להלן לשגרת האירוע Click של הלחצן cmdState:

```
Dim sState As String
Dim sSql As String

'BUILD SQL QUERY
sSql = "Select * from Publishers"
sState = InputBox$("Enter State Abbreviation:")
If sState <>"" Then sSQL = sSQL & " Where State = '"& sState & "'"

'UPDATE THE ADO DATA CONTROL
Adodc1.CommandType = adCmdText
Adodc1.RecordSource = sSQL
Adodc1.Refresh
```

הפעל שוב את פרויקט הדוגמה ולחץ על הלחצן שהוספת. הקלד NY, כדי להציג רק רשומות של מוציאים לאור הפועלים במדינת ניו-יורק. תוכל לוודא את הצלחת הפעולה על ידי לחיצה על לחצני הניווט.

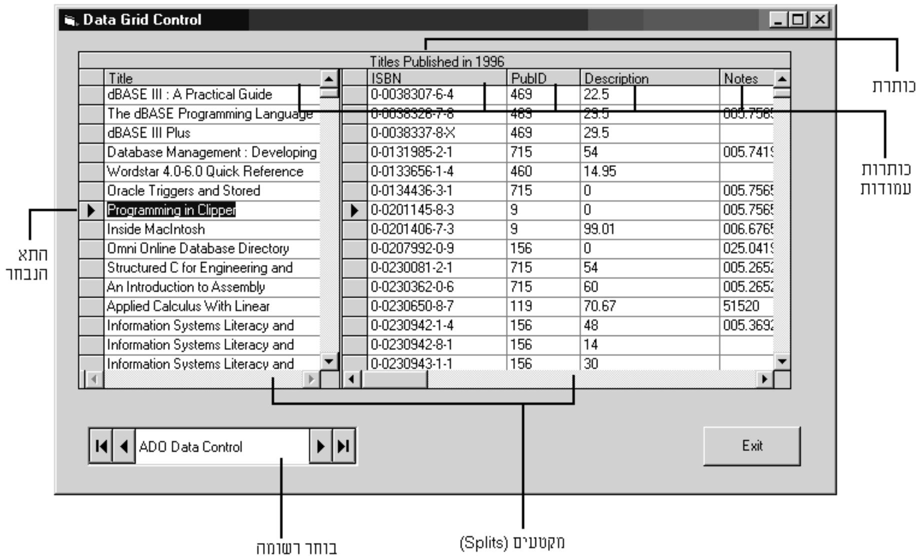
שימוש בפקד DataGrid

חלק ניכר מהנתונים שמשמשים לעבודה עם תוכניות מחשב מוצגים במבנה **רשת** (Grid) שמורכבת מעמודות ושורות, כדוגמת הנתונים המוצגים בגיליון אלקטרוני של Excel. מבנה רשת הוא מבנה התצוגה המועדף עבור משתמשים רבים ועבור סוגים נתונים שונים. הפקד DataGrid, שהוא פקד חדש שנכלל לראשונה בגרסת Visual Basic 6, מאפשר לך להציג נתונים הכלולים במסד הנתונים שלך תוך שימוש במבנה רשת. במצבים מסוימים אף מתאפשר למשתמש לבצע גישה ישירה לתאים מסוימים ברשת, ולשנות את המידע המוצג בהם.

ייתכן שיצא לך לעבוד עם פקד DBGrid שנכלל בגרסאות הקודמות של Visual Basic. אך בשונה מפקד DBGrid, פקד DataGrid נועד לעבוד עם אובייקטי ADO. פקד DataGrid קשור לפקד ADO Data שתואר בתחילת הפרק, במקום לפקד Data מיושן.

בתרשים 28.9 מוצגת דוגמה לשימוש ב-DataGrid. הפקד משמש להצגת הטבלה Titles שבמסד הנתונים BIBLIO שמסופק יחד עם Visual Basic.

פקד DataGrid נוח לשימוש וזאת למרות שתידרש להבין את אופן פעולתו לקשר אותו למסד הנתונים. אך לפני שיתאפשר לך להשתמש בפקד, תידרש להוסיף אותו לארגז הכלים. הוסף אותו על ידי לחיצה ימנית על שטח ריק בארגז הכלים, בחירה בפקודה **Components** מהתפריט תלוי ההקשר שיוצג וסימון ליד הפריט Microsoft DataGrid Version 6.0 בתיבת הדו-שיח. לחץ על OK כדי לסגור את תיבת הדו-שיח.



תרשים 28.9: פקד DataGrid משתמש במערך רשומות שמספק פקד ADO Data

הזנת נתונים לרשת

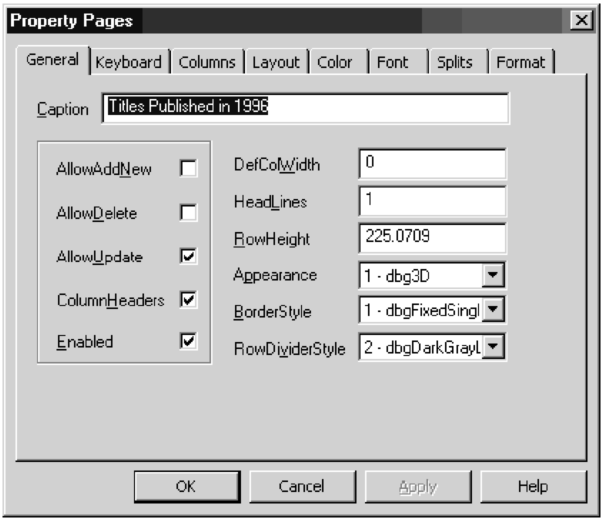
כדי שתוכל להשתמש בפקד DataGrid תידרש להגדיר עבורו את המבנה הרצוי לך ולחבר אותו למקור נתונים. חיבור הפקד למקור הנתונים מתבצע באותו אופן בו מקשרים פקדי איגוד נתונים אחרים למסדי נתונים, על ידי ביצוע הפעולות שלהלן:

1. שרטט פקד ADO Data על הטופס והגדר אותו באופן שתואר בסעיף **שימוש בפקד ADO Data**.
2. הצג את חלון המאפיינים של פקד DataGrid והצב במאפיין DataSource את שם פקד ADO Data (בוודאי הבחנת שלפקד DataGrid אין מאפיין DataField, זאת היות ופקד DataGrid יכול להציג כמה שדות בו-זמנית).
3. פקד DataGrid יכול להגדיר את מספר העמודות שבו באופן אוטומטי, בהתאם למבנה הרשומות שבמקור הנתונים אליו הוא מקושר. לשם כך יהיה עליך ללחוץ לחיצה ימנית על הפקד ולבחור את הפקודה Retrieve Fields מהתפריט תלוי ההקשר שיוצג. מספר העמודות ברשת יותאם למספר השדות שבמערך הרשומות של פקד ADO Data.
4. התאם את עיצוב הרשת לצרכיך ובצע פעולות עיצוב שאותן יש לבצע באופן ידני.

ראה "היכרות עם פקדי איגוד נתונים", פרק 25.

הגדרת פקד DataGrid

פקד DataGrid הוא פקד מורכב יחסית שמספר גדול של מאפיינים משמשים להגדרת פעולתו. בדומה לפקדים אחרים, גם לפקד DataGrid דפי מאפיינים המאפשרים גישה מסודרת למאפייניו. אם תרצה להציג את דפי המאפיינים של הפקד, לחץ לחיצה ימנית על הפקד ובחר את הפקודה Properties מהתפריט תלוי ההקשר שיוצג. תיבת הדו-שיח Property Pages של פקד DataGrid מוצגת בתרשים 28.10.



תרשים 28.10: את רוב הגורמים הקשורים בפעולת פקד DataGrid ניתן להגדיר על ידי שימוש בדפי המאפיינים שלו

בסעיפים הבאים נבחן את אופן השימוש בכל אחד מדפי המאפיינים לשם שליטה על פקד DataGrid ולשם בחינת הנושאים הקשורים בחיבור הפקד למקור נתונים.

הכרטיסיה General

הכרטיסיה General המוצגת בתרשים 28.10 כוללת כמה מאפיינים שמגדירים את אופן הפעולה הכללי של הפקד. כרטיסיות אחרות, כגון Columns ו-Layout, מאפשרות לך להגדיר את המאפיינים של מבנה הרשת.

הערה:

מידע אודות הכרטיסיות Columns ו-Layout תוכל למצוא בסעיף **התאמה אישית של מראה פקד DataGrid**. אפשרויות נוספות לעיצוב הפקד תתוארנה בסעיפים **התאמה אישית של פקד DataGrid באמצעות קוד ופיצול פקד DataGrid**.

להלן מפורטים כמה מאפיינים אשר מגדירים את חזות הפקד:

- מאפיין Caption שולט על הכותרת שתוצג בשוליים העליונים של הרשת. אם אינך מעוניין להציג כותרת, הצב במאפיין מחרוזת ריקה.
- מאפיין ColumnHeader הוא שקובע האם תוצג בקצה העליון של הרשת שורת כותרות העמודות. כותרת ברירת המחדל של כל עמודה היא השם שהוגדר עבור השדה התואם לו במסד הנתונים, אך בהמשך תלמד כיצד תוכל לשנות כותרות כדי להתאימן לצרכיך. גובה שורת הכותרות יוגדר באמצעות המאפיין HeadLines. הצב במאפיין זה ערך שלם שייצג את הגובה הרצוי לך.
- ניתן לשלוט על המראה הכללי של הרשת באמצעות המאפיין Appearance. בשורת ההגדרה של המאפיין תמצא שתי מחרוזות: dbgFlat ו-dbg3d (מחרוזת ברירת המחדל), אשר גורמות לרשת להיראות שטוחה או תלת-ממדית, בהתאמה.
- מאפיין RowDividerStyle מגדיר את סוג המפריד בין שורות נתונים. תוכל לבחור את הסוג הרצוי מבין כמה אפשרויות, ביניהן אפשרות שלא להציג מפרידים. מאפיין RowDividerStyle אינו משפיע על המפרידים של בוררי הרשומות.
- מאפיין Enabled הוא שקובע האם יתאפשר למשתמש לקיים אינטראקציה עם פקד DataGrid. אם תציב במאפיין ערך False, לא יתאפשר למשתמש לגלול את הרשת, לבחור תאים או לשנות נתונים שיוצגו. אך גם במצב כזה הפקד וחלק מהנתונים הכלולים בו יוצגו על הטופס.

בנוסף למאפיינים המשמשים להגדרת חזות הפקד, כלולים בכרטיסיה General גם שלושת המאפיינים המפורטים להלן אשר מגדירים אילו פעולות יתאפשר לך לבצע על הנתונים שבפקד DataGrid:

- AllowAddNew - קובע האם יתאפשר למשתמש להוסיף רשומה למערך הרשומות המוצג במסגרת פקד DataGrid (הגדרת ברירת המחדל של הפקד היא False). אם המאפיין מוגדר True, תוצג שורה ריקה בסוף מערך הרשומות, כדי לאפשר למשתמש להזין נתונים. בבורר הרשומות של שורת ההוספה תוצג כוכבית (*).
- AllowDelete - קובע האם יתאפשר למשתמש לגרוע רשומות ממערך הרשומות בו משתמש הפקד. אם תרצה לגרוע רשומה, היעזר בבורר הרשומות כדי לסמן אותה והקש על Delete. הגדרת ברירת המחדל של המאפיין AllowDelete היא False.
- AllowUpdate - קובע האם יתאפשר למשתמש לערוך רשומות קיימות (הגדרת ברירת המחדל של המאפיין היא True). הצבת ערך False במאפיין גורמת לתאים שבפקד DataGrid לפעול כמו תיבות טקסט נעולות: יתאפשר לך לבחור את הטקסט בעזרת העכבר ולהעתיק אותו ללוח, אך לא תוכל לשנות אותו.

את שלושת המאפיינים האלה ניתן להגדיר בעת עיצוב היישום וגם בזמן ריצה. אם תציב ערכי False בשלושתם, פקד זה יהיה פקד לקריאה בלבד.

אפשרויות הקשורות בשימוש במקלדת

ניתן אומנם לבחור תאים ברשת בעזרת העכבר, אך ייתכן שתעדיף להשתמש במקלדת, היות וכך יתאפשר לך לבצע פעולות מהר יותר. פקד DataGrid כולל מספר מאפיינים אשר מגדירים את אופן פעולת המקלדת במסגרת העבודה עמו. תוכל להגדיר את המאפיינים האלה על ידי שימוש בכרטיסיה Keyboard בגיליון המאפיינים של הפקד, או על ידי שימוש בקוד.

לדוגמה, המאפיין AllowArrows קובע האם ניתן יהיה להשתמש במקשי החיצים למעבר בין תאים בפקד DataGrid. אם תציב ערך False במאפיין זה, המשתמש ייאלץ להיעזר במקש Tab או בעכבר לשם מעבר בין תאי פקד Grid.

הערה:

בעת עריכת תאים, מקשי החיצים משמשים לתנועה בתחומי התא אותו אתה עורך כעת. אם תרצה להיעזר בהם לתנועה בין תאים ברשת, לחץ פעם אחת על תא מסוים כדי להאיר אותו וכך יתאפשר לך לעבור בין תאים.

הגדרת אופן השימוש במקש Tab בפקד כלשהו בעל מבנה רשת, מהווה דילמה עבור מפתחים, מפני שבמצב רגיל מקש זה משמש להעברת מוקד הקלט לפקד הבא בטופס. למרבה המזל, מאפיין TabAction של פקד DataGrid יאפשר להגדיר את אופן פעולת המקש כרצוי לך. מאפיין זה יכול לקבל אחד משלושה ערכים אפשריים:

- dbgControlNavigation - 0 - הקשה על Tab בעת עבודה עם פקד DataGrid תעביר את מוקד הקלט לפקד הבא בטופס. הגדרה זו גורמת לפקד DataGrid לנהוג כמו כל פקד רגיל אחר.
- dbgColumnNavigation - 1 - הקשה על מקש Tab מעבירה לתא הבא בשורה הנוכחית של פקד DataGrid. הקשה על Shift+Tab מעבירה לתא הקודם בשורה. הקשה על מקש Tab כאשר הסמן מצוי בתא האחרון בשורה תעביר אותך לפקד הבא בטופס. למעבר לשורה אחרת בפקד DataGrid יהיה עליך להשתמש במקשי החיצים או בעכבר.
- dbgGridNavigation - 2 - הגדרה זו אינה שונה במהותה מההגדרה הקודמת, אך הבחירה בה תאפשר להגדיר שני מאפיינים נוספים השולטים על מקש Tab: TabAcrossSplits ו-WrapCellPointer. ערך True במאפיין WrapCellPointer יאפשר להיעזר במקש Tab לשם מעבר בין שורות בפקד DataGrid. אם תגדיר את המאפיין TabAcrossSplits לערך True, תוכל להשתמש במקש Tab כדי לעבור בין מקטעים בפקד DataGrid.

פיצול הרשת

ברוב המקרים, פקד DataGrid מתבסס על סדרת נתונים אחת, אך ניתן לחלק את הרשת למספר חלקים. חלקים אלה, המכונים **מקטעים** (Splits), מתפקדים במידה רבה כרשתות משנה של הרשת הכללית. בתרשים 28.9 תוכל לראות דוגמה לשימוש במקטעים. השדה Title (כותרת) כלול במקטע נפרד משאר השדות ועובדה זו מאפשרת לו להיות גלוי על המסך בעת גלילה אופקית של מקטעים אחרים.

הערה:
גם אם פקד DataGrid מכיל מספר מקטעים, עדיין קיימת רק רשומה נוכחית אחת, וכל המקטעים מכילים את אותם נתונים מאותו פקד ADO. השימוש במקטעים אינו כה מורכב כפי שעשוי להיראות בתחילה. תוכל להתייחס למקטעים כאל קטעי רשת שאת כל אחד מהם ניתן לעצב בעיצוב שונה.

עריכת הרשת בעת עיצוב היישום

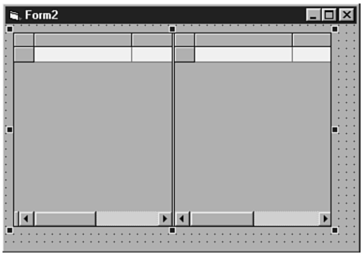
אם תרצה להוסיף או לגרוע מקטעים במסגרת עיצוב היישום, תצטרך לעבור למצב עריכת עיצוב סביבת הרשת. כדי לעבור למצב זה, לחץ לחיצה ימנית על הפקד, כפי שעשית כדי להציג את גיליון המאפיינים, אך במקום לבחור מהתפריט תלוי ההקשר את הפקודה Properties, בחר את הפקודה Edit. לרגע ייראה לך שהפקודה לא עשתה כלום, אך אם תלחץ שוב לחיצה ימנית תראה שהפריטים שבתפריט תלוי ההקשר השתנו מתפריט Visual Basic לתפריט פקד DataGrid. כאשר תעבוד במצב זה, התפריט תלוי ההקשר יאפשר לך לבצע את הפעולות הבאות:

- להוסיף ולגרוע עמודות.
- לשנות גודל עמודות.
- לאחזר את הגדרת מבנה השדות של מקור הנתונים.
- לבטל הגדרות עיצוב קיימות של השדות.
- להוסיף ולגרוע מקטעים.



יצירת מקטע חדש

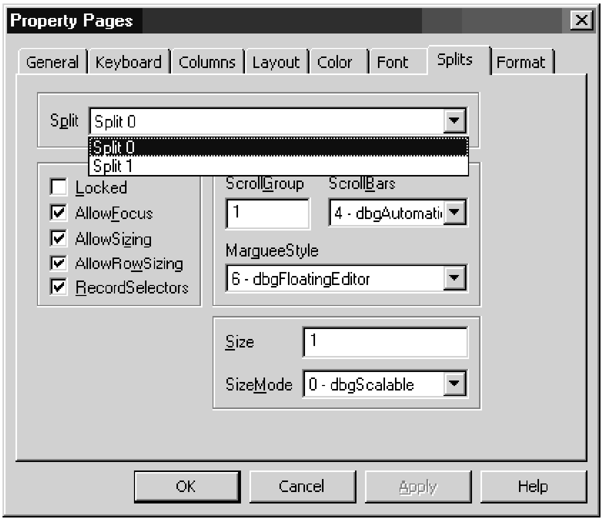
כברירת מחדל, פקד DataGrid מכיל רק מקטע אחד שנקרא Split0, כמתואר בתרשים 28.11. אם תרצה להוסיף לפקד DataGrid מקטע, לחץ לחיצה ימנית על הפקד בעת עבודה במצב עיצוב הרשת ובחר מהתפריט תלוי ההקשר את הפקודה Split. פעולה זו תוסיף לפקד מקטע נוסף, כמתואר בתרשים 28.11.



תרשים 28.11:
פקד DataGrid לפני הוספת מקטע נוסף ואחריו

עבודה עם מאפייני מקטעים

בכל פעם שתוסיף לפקד DataGrid מקטע, המקטע החדש יתוסף לרשימה הנפתחת Split שבתיבת הדו-שיח Property Pages של הפקד, כמתואר בתרשים 28.12.



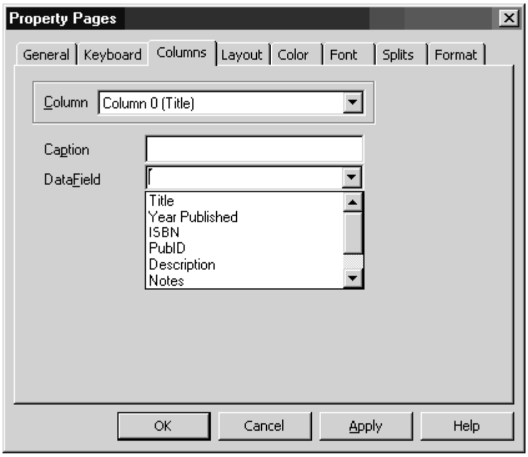
תרשים 28.12: הכרטיסיה Splits שבתיבת הדו-שיח Property Pages של פקד DataGrid מאפשרת להגדיר מאפייני מקטעים של הפקד

בעת יצירת מקטעים חדשים מוגדרים עבורם שמות סדרתיים: Split0, Split1, Split2 וכך הלאה. אם תרצה לקבוע הגדרות למקטע מסוים, בחר אותו מהרשימה הנפתחת Split והגדר את המאפיינים באופן הרצוי לך. להלן כמה מבין המאפיינים הרלוונטיים:

- Locked - בחירה במאפיין זה מונעת מהמשתמש להזין טקסט.
- AllowFocus - הצבת ערך False במאפיין זה תמנע מהמקטע הנוכחי לקבל מוקד.
- AllowSizing - מאפשר למשתמש לשנות את רוחב המקטע תוך כדי פעולת התוכנית.
- AllowRowSizing - מאפיין זה יכול למנוע מהמשתמש לשנות את גודל השורות בזמן ריצה.
- RecordSelectors - המאפיין מאפשר או מנטרל בוררי רשומות של המקטע הנוכחי. אם תגדיר את מאפיין RecordSelectors של הפקד, ההגדרה העדכנית תחול על המקטע הנוכחי.

התאמת עיצוב הפקד

אומנם היכולת להגדרה אוטומטית של תצורת הפקד הכלולה בפקד DataGrid היא כלי נחמד ומועיל, אך במקרים רבים תרצה לשנות את מאפייני תצורת ברירת המחדל. לדוגמה, ייתכן שתרצה להגדיר שני מקטעים שיכללו עמודות שונות. אם תרצה לנהוג כך תידרש תחילה להגדיר את המקטעים ואחר כך להיעזר בכרטיסיות Layout ו-abs שבגיליון המאפיינים של הפקד, כדי להתאימם לצרכיך. הכרטיסיה Columns שולטת על סדר הצגת העמודות בפקד (ראה תרשים 28.13).



תרשים 28.13: אם תרצה תוכל לשנות את סדר הצגת העמודות בפקד באופן ידני

קל לעבוד עם הכרטיסיה Columns. תידרש בסך הכל לבחור עמודה ולבחור את המאפיין Caption ואת שם שדה הנתונים.

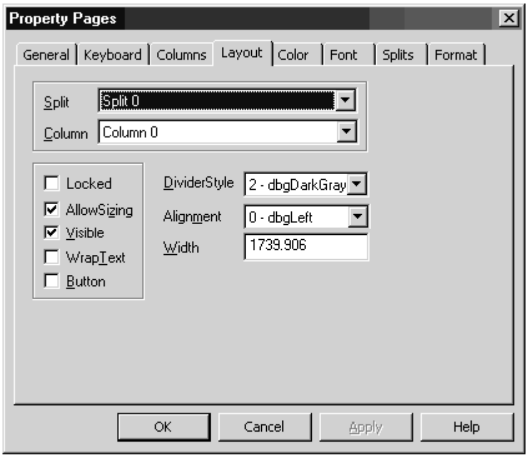
הערה:

אם לא תזין ביטוי חוקי עבור המאפיין DataField, העמודה תהיה ריקה.

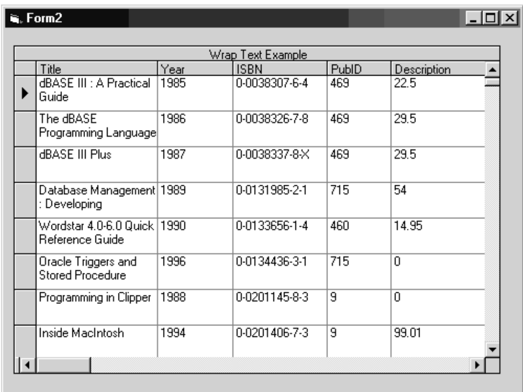
הכרטיסיה Tabs שולטת על העמודות שתוצגנה ואילו הכרטיסיה Layout שולטת על עיצוב כל עמודה בנפרד. אם תרצה להשתמש בכרטיסיה Layout המוצגת בתרשים 28.14, תידרש לבחור מקטע ועמודה, ואחר תידרש להגדיר את המאפיינים הבאים :

- Locked - כאשר המאפיין הזה מוגדר True, הוא מונע עריכת עמודה, גם אם ההגדרות הכלליות של הפקד מאפשרות זאת.
- AllowSizing - מאפיין זה, שערך ברירת המחדל שלו הוא True, מאפשר למשתמש לשנות רוחב עמודות בעת עיצוב היישום.
- Visible - מאפיין זה קובע האם העמודה תוצג בפקד (אך חשוב גם שתזכור שאם אינך צריך להציג נתונים בעמודה מסוימת, תוכל למנוע את הצגתה על ידי גריעת העמודה ממקור הנתונים של הפקד).

- WrapText - מאפיין זה קובע האם הטקסט בעמודה יוצג כשהוא פרוש על כמה שורות ואם רוחבו יהיה גדול מרוחבה. בתרשים 28.15 תוכל לראות דוגמה לשימוש במאפיין WrapText.
- Button - מאפיין זה ייקבע האם יוצג לחצן רשימה נפתחת במסגרת השדה המואר של הפקד.
- DividerStyle - מאפיין זה דומה למאפיין RowDividerStyle שבכרטיסייה General של תיבת הדו-שיח, אך ניתן להגדיר אותו עבור כל עמודה בנפרד.
- Alignment - מאפיין זה שולט על אופן יישור הערכים שיוצגו במסגרת העמודה.
- Width - מאפיין זה מגדיר רוחב עמודה מסוימת (דרך פשוטה יותר להגדרת רוחב העמודה היא על ידי שימוש באופן העריכה שתואר קודם לכן ושימוש בעכבר להגדרת הרוחב).



תרשים 28.14: כרטיסיית Layout שבגיליון המאפיינים של פקד DataGrid



תרשים 28.15: בדוגמה זו, מאפיין WrapText של המקטע השמאלי מוגדר True

התאמת הרשת על ידי קוד

במהלך הלימוד על פקד DataGrid, התרכזת בהגדרת מאפיינים בסביבת העיצוב. אך כמעט כל מה שלמדת ניתן ליישום גם בסביבת זמן ריצה. לדוגמה, כדי להחיל את המאפיין WrapText על העמודה הראשונה של המקטע הראשון, השתמש בקוד הבא:

```
datagrid1.Splits(0).Columns(0).WrapText = True
```

אומנם בחלק מהמקרים תידרש לעבור דרך מספר רמות אוספים כדי להגיע למאפיין הרצוי לך, אך ניתן לשנות את רוב המאפיינים של פקד DataGrid באמצעות קוד.

שימוש באובייקטי ActiveX Data

פקד ADO Data Control שימושי רק עד גבול מסוים. במה שנוגע לטיפול בנתונים, את העוצמה האמיתית של Visual Basic תוכל לחוש כאשר תשתמש באובייקטי ADO. האובייקטים המרכיבים את מודל האובייקטים של ADO יפורטו בטבלה 28.1.

טבלה 28.1: האובייקטים של טכנולוגיית ActiveX Data Objects

אובייקט	תיאור
RecordSet	מכיל את כל הרשומות המהוות תוצאה של שאילתה
Connection	מאפשר שליטה על החיבור למסד הנתונים
Command	מבצע פקודות של מסד הנתונים ושאילתות, תוך שימוש בשאילתות עם פרמטרים (Parameterized Queries)
Error	מאחזר שגיאות של ADO
Field	מייצג פיסת מידע במערך רשומות
Parameter	פועל בשילוב עם אובייקט Command כדי להגדיר פרמטר שאילתה או שיגרה מאוחסנת
Property	מאפשר למשתמש גישה למאפייני אובייקט ADO

כדי להשתמש באובייקטי ADO, כלול בפרויקט הפניה מתאימה. בחר **References** מתפריט **Project**, סמן ליד **Microsoft ActiveX Objects 2.0** ולחץ על OK. לאחר שתוסיף את ההפניה, תוכל להשתמש בתחיליות שלהלן, בהצהרות על אובייקטי ADO:

- ADODB - ספריה זו כוללת את כל האובייקטים שפורטו בטבלה 28.1.
- ADOR - ספריה זו כוללת את כל האובייקטים הקשורים בעבודה עם Recordsets (Record, Field, Property). הספריה מסופקת עם Internet Explorer ומאפשרת למפתחים להעביר **מערכי רשומות בלתי מקושרים** (Disconnected Recordsets) באמצעות ה-Web.

במהלך הסעיפים הבאים נבחן חלק מהמאפיינים, השיטות והאירועים של אובייקטי ADO. רשימה מלאה של הגורמים האלה ניתן למצוא ב- Object Browser, הניתן להצגה באמצעות הקשה על F2.

שימוש ב-ADO ליצירת חיבורים

אובייקט Connection משמש להגדרת חיבור למקור נתונים. המאפיין החשוב ביותר שלו הוא **ConnectionString**, אשר מכיל את הנתונים הדרושים לאובייקט לשם התחברות למסד הנתונים. תוכל לשלוט על מצבו של אובייקט Connection בעזרת השיטות Open ו-Close.

פתיחה וסגירת חיבור

הצעד הראשון בפתיחת חיבור למסד נתונים היא יצירת מופע חדש של אובייקט ADODB.Connection, באופן הבא:

```
Dim cn As ADODB.Connection
Set cn = New ADODB.Connection
```

לאחר יצירת המופע החדש של האובייקט, כל שתידרש לעשות כדי להגדיר חיבור הוא להגדיר ***מחרוזת חיבור*** (Connection String) ולקרוא לשיטה Open. תוכל להגדיר את מחרוזת החיבור בשני אופנים. האחד, על ידי הגדרת מאפיין ConnectionString של אובייקט Connection:

```
cn.ConnectionString = "DSN-BIBLIO"
cn.Open
```

באופן השני, העבר את המחרוזת כחלק מהקריאה לשיטת Open של האובייקט:

```
cn.Open "DSN-BIBLIO"
```

הערה:

מחרוזת החיבור שהוצגה בדוגמה היא פשוטה. בהתחברות למקורות נתונים אחרים ייתכן שתידרש למחרוזות חיבור ארוכות יותר אשר תכלולנה נתונים נוספים.

לאחר שתסיים להשתמש באובייקט Connection, קרא לשיטה Close. בדומה לאופן השימוש באובייקטים מסוגים אחרים, גם בעת סגירת אובייקט זה כדאי להציב ערך Nothing באובייקט כדי לשחרר את משאבי המערכת ששימשו אותו.

```
cn.Close
Set cn = Nothing
```

שימוש בשיטה Execute

ADO מאפשר להשיג מטרות מסוימות תוך יישום אופני פעולה שונים. אחזור נתונים הוא דוגמה טובה לכך. לביצוע פעולות לאחזור נתונים תוכל לבחור באובייקטים Recordset, Command או Connection, שלכולם שיטה המאפשרת לאחזר נתונים לתוך מערך רשומות. השיטה Execute של אובייקט Connection מאפשרת להפעיל משפט SQL על מקור נתונים. אם משפט SQL יחזיר רשומות, תוכל לגשת אליהן פשוט על ידי הצבת ערך שיוחזר על ידי השיטה Execute באובייקט ADO Recordset.

נסה את השיטה Execute בדוגמה פשוטה. התחל פרויקט חדש מסוג Standard EXE והוסף לו הפניה לספריית ADO. הוסף לטופס פקד ListBox וקרא לו lstAuthors. לבסוף, הוסף את שורות הקוד הבאות לשגרת האירוע Load של הטופס:

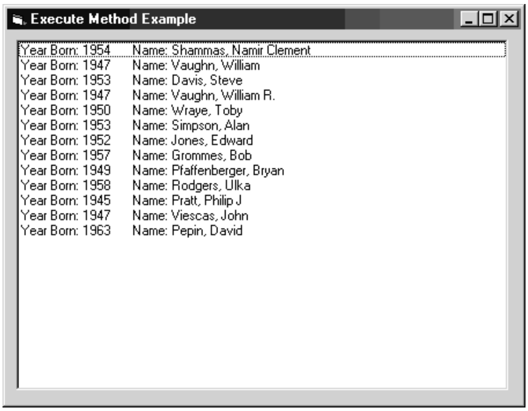
```
Dim cn As ADODB.Connection
Dim rs As ADODB.Recordset

Set cn = New ADODB.Connection
cn.Open "DSN=BIBLIO"

Set rs = cn.Execute("Select * from Authors where [Year Born] > 1943")
Do While Not rs.EOF
        lstAuthors.AddItem "Year Born: " & rs.Fields("Year Born") & vbTab & "Name: " & _
                    rs.Fields("Author")
        rs.MoveNext
Loop

rs.Close
cn.Close
Set rs = Nothing
Set cn = Nothing
```

הפעל את התוכנית. תוצגנה לפניך רשומות כמו אלו המוצגות בתרשים 28.16.



תרשים 28.16: תוצאות הפעלת שאילתה פשוטה המוצגות כאן נוצרו על ידי שימוש בשיטה Execute של אובייקט ADODB.Connection

השיטה Execute של אובייקט Connection מחזירה אובייקט מסוג Recordset שמאוחסן במשתנה rs. אחר נעשה שימוש בלולאת While למעבר על מערך הרשומות והוספת שם כל מחבר ושנת לידתו לתיבת הרשימה.

תוכל להיעזר בשיטה Execute גם להפעלת משפטי SQL שלא יחזירו מערכי רשומות, כמו בדוגמאות הבאות:

```
cn.Execute "Delete from Authors where Author = ' Jones, Amanda'"
cn.Execute "INSERT INTO AUTHORS ([Author], [Year Born]) Select 'Siler, Brian', 1972"
```

משפט SQL הראשון שהוצג כאן יגרע רשומה מטבלת Authors ואילו המשפט השני יוסיף רשומה למערך הרשומות (זכור שמשפטי הדוגמה נכתבו כך שיעבדו עם Access, בשימוש במסדי נתונים אחרים תחביר משפטי SQL עשוי להיות מעט שונה).

התחברות למסד נתונים ללא שימוש ב-DSN

עד כה השתמשנו בפרק זה במחרוזת חיבור פשוטה שכוללת את שם מקור הנתונים שהוגדר על בסיס מסד הנתונים BIBLIO, והוכן לפעולה בעזרת ODBC Data Source Manager. אם לא תרצה להיות תלוי בהגדרות DSN's, תוכל להגדיר חיבור שלא יהיה מבוסס על DSN (DSN-less Connection), על ידי כך שתכלול במחרוזת ConnectionString נתונים נוספים אשר דרושים ל-ADO, כפי שנעשה בדוגמאות הבאות אשר מתייחסות למסד נתונים של Access:

```
cn.ConnectionString ="Provider=Microsoft.Jet.OLEDB.3.51;Data Source=d:\vb6\biblio.mdb"

cn.ConnectionString = "Driver=Microsoft Access Driver (*.mdb) ;DBQ=d:\vb6\biblio.mdb"
```

הערה:

גם אם תגדיר חיבור שלא יתבסס על DSN, עדיין תצטרך שהקבצים ומנהלי ההתקן הרלוונטיים יהיו מותקנים על מחשב המשתמש.

מחרוזות החיבור שבדוגמאות שהצגנו כללו פניות למנהלי ההתקן, במקום ל-DSN's. בשימוש במנהל ההתקן של Access, פרמטר DBQ מצביע על שם מסד נתונים. תוכל להגדיר גם OLE DB Providers אחרים, כדוגמת המחרוזת הבאה, אשר משמשת להתחברות ל- SQL Server:

```
"Provider=SQLOLEDB.1;USERID= apower;Password=groovy:InitialCatalog=DB; _
        DataSource=Server1"
```

בדוגמה האחרונה יכולת לראות שמחרוזת חיבור יכולה לכלול בתוכה נתונים רלוונטיים רבים ביניהם שם משתמש, סיסמה ושם ברירת מחדל של מסד הנתונים.

עבודה עם מערכי רשומות

מערכי רשומות (Recordsets) מייצגים נתונים אמיתיים במסד נתונים. מערכי רשומות מכילים שדות (כגון שם ומספר טלפון) וערכים לאותם שדות (למשל John Smith ו-555-1111). כל סדרת ערכי שדות הקשורים אחד לשני מרכיבה רשומה (Record) וכל הרשומות יחד מרכיבות מערך רשומות.

דרך פשוטה לתאר מערך רשומות באופן מוחשי היא על ידי שימוש בגיליון אלקטרוני או בפקד Grid. השדות הם עמודות והשורות הן רשומות. רוב מערכי הרשומות נוצרים כתוצאה מהפעלת שאילתה על מסד נתונים. ניתן להשתמש במערכי רשומות בתוכניות לשם אחזור ועדכון נתונים.

בעת שימוש ב-ADO, מערך רשומות מאוחסן ב**אובייקט Recordset**. לאחר שאובייקט Recordset מאוכלס בנתונים ניתן לבצע עליהם את הפעולות שלהלן:

- הוספת רשומות חדשות.
- עריכת רשומות קיימות.
- גריעת רשומות.
- ניווט במערך הרשומות (שינוי הרשומה הנוכחית).

יצירת מערך רשומות

למדת כיצד ליצור מערך רשומות ולאכלס אותו בנתונים באמצעות שיטה Execute של אובייקט Connection. אך לאובייקט Recordset יש שיטות ומאפיינים משלו שיכולים לשמש לאחזור נתונים. כפי שקורה בכל האובייקטים, כדי שתוכל להשתמש במאפיינים אלה תידרש תחילה ליצור מופע חדש של אובייקט Recordset:

```
Dim rs As ADODB.Recordset
Set rs = New ADODB.Recordset
```

לאחר מכן, תוכל להיעזר במאפייני האובייקט לצורך הגדרת חיבור, מקור רשומות וסוג מערך רשומות.

להגדרת מקור נתונים של אובייקט Recordset, הצב במאפיין ActiveConnection של האובייקט את שם אובייקט ADO Connection

```
rs.ActiveConnection = cn
```

שורת קוד זו התבססה על ההנחה כי cn הוא שם חיבור הפתוח כעת אשר מצביע על מקור נתונים, כפי שתואר קודם לכן. דרך נוספת בה ניתן לבחור בחיבור מסוים היא הצבת מחרוזת חיבור במאפיין ActiveConnection.

```
rs.ActiveConnection = "DSN=BIBLIO"
```

אם תשתמש במחרוזת, במקום באובייקט Connection, אובייקט Recordset יפתח חיבור משלו למסד הנתונים.

השיטה Open של אובייקט Recordset, תגרום לאכלוס מערך הרשומות בנתונים. כדי שתוכל להשתמש בשיטה Open, תידרש תחילה להגדיר את מאפיין Source של האובייקט, שמקביל במידה מסוימת למאפיין RecordSource של פקד ADO Data. הקוד המוצג בתוכנית 28.1 ייצור אובייקט Recordset חדש וידפיס את הנתונים.

תוכנית 28.1: ADOTEST.VBP - **יצירת מערך רשומות תוך שימוש ב-ADO.**

```
Dim sSQL As String
Dim cn As ADODB.Connection
Dim rs As ADODB.Recordset

'Const sSQL =sSQL = "SELECT * FROM Authors Where [Author] Like '" & txtLetter & "*'"

'OPEN A CONNECTION
Set cn = New ADODB.Connection
cn.Open "DSN=BIBLIO"

'OPEN A RECORDSET
Set rs = New ADODB.Recordset
rs.ActiveConnection = cn
rs.Source = sSQL
rs.Open

'PRINT THE AUTHORS' NAMES
rs.MoveFirst
Do While Not rs.EOF
        Debug.Print "author = " & rs.Fields("Author")
        rs.MoveNext
Loop

'CLOSE EVERYTHING
rs.Close
Set rs = Nothing
cn.Close
Set cn = Nothing
```

קטע הקוד בתוכנית 28.1 יפעיל שאילתת SQL שתבחר את כל המחברים ששם משפחתם מתחיל באות Q.

הערה:

תחביר הביטוי 'Q*' Like של משפט SQL ייחודי ל-Access. בסביבת SQL Server, המשפט הזה היה נכתב כך:

```
Select * from Authors where Author like "Q*"
```

הצגת ערכי שדות

כדי לגשת לנתונים המאוחסנים באובייקט Recordset, תוכל להסתייע באוסף Fields. טבלה 28.2 תפרט אפשרויות שבהן תוכל להשתמש לגישה לשדה מסוים באובייקט.

טבלה 28.2: עבודה עם ערכי שדות

מבנה תחבירי	דוגמה
Recordset.fields("field name")	rsPeople.Fields("LastName")
Recordset.fields(index)	rsPeopleFields(2)
Recordset![field name]	rsPeople![Last Name] rsPeople!LastName

בדוגמת הקוד שתובא כעת, נשתמש בתחביר Recordset.Fields("field name"), אך במקרים מסוימים (למשל כאשר שם השדה הרצוי אינו ידוע לך), עדיף להשתמש באינדקס השדה. אינדקסי שדות מתחילים באפס ומסתיימים בערך הקטן ב-1 ממספר השדות שבמערך הרשומות. את מספר השדות במערך הרשומות תוכל לברר באמצעות המאפיין Count של האוסף Fields. קטע הקוד הבא ידפיס שם כל שדה ויעבור על מערך הרשומות תוך הדפסת ערך כל שדה:

```
'PRINT THE FIELD NAMES
For I = 0 To rs.Fields.Count - 1
        Debug.Print rs.Fields(I).Name
Next I
Debug.Print

'PRINT CONTENTS OF ALL FIELDS IN EACH RECORD
rs.MoveFirst
Do While Not rs.EOF
        For I = 0 To rs.Fields.Count - 1
                Debug.Print rs.Fields(I),
        Next I
        rs.MoveNext
Loop
```

קטע הקוד הזה משתמש בשני מאפיינים של אובייקט Field: Name ו-Value. שים לב שבעת התייחסות למאפיין Value לא תידרש לציין את שמו מפני שזהו מאפיין ברירת המחדל של האובייקט.

ניווט במערכי רשומות

לאחר שתייבא את הנתונים לאובייקט Recordset, תוכל לבצע גישה לשדות הרשומה הנוכחית ולעדכן את ערכיהם. תאר לעצמך את מערך הרשומות כקובץ סדרתי ארוך. בכל רגע נתון, הרשומה הנוכחית מכילה מצביע למקום כלשהו בקובץ. אם תרצה לעבור לרשומות אחרות תוכל להיעזר בשיטות הבאות:

- MoveFirst - עוברת לרשומה הראשונה, הרשומה המצויה מייד לאחר סמן תחילת הקובץ (Beginning Of File - BOF).
- MoveLast - עוברת לרשומה האחרונה, זו המצויה ממש לפני סמן סוף הקובץ (End Of File - EOF).
- MoveNext - עוברת לרשומה שאחרי הרשומה הנוכחית כעת (כלומר נעה לכיוון סמן EOF).
- MovePrevious - עוברת לרשומה שלפני הרשומה הנוכחית כעת (כלומר נעה לכיוון סמן BOF).
- Move - מעבירה את הסמן מספר נתון של רשומות, קדימה או אחורה.

BOF ו-EOF הם מאפיינים המצביעים על התחלה וסיום הקובץ, בהתאמה.

ניווט והמאפיין CursorType

בדוגמאות שהבאנו עד כה הוצגו רק שתי שיטות שמשמשות לניווט במסגרת מערך רשומות: MoveFirst ו-MoveNext. בתוכנית 28.1 נעזרת בלולאת Do While כדי לעבור על הרשומות שבמערך הרשומות הפתוח, תוך כדי תנועה קדימה.

אם היית מנסה להשתמש בשיטות MovePrevious ו-MoveLast בקטע הקוד שבתוכנית 28.1 היתה נגרמת שגיאה, מפני שלא ניתן להשתמש בשיטות אלו בעבודה עם המאפיין Cursor Type ששימש בקטע קוד זה. סמן במערך רשומות דומה לסמן המוצג על מסך המחשב בכך ששניהם מצביעים על המיקום הנוכחי.

משום שקטע הקוד לא כלל הגדרת ערך למאפיין CursorType של אובייקט RecordSet, נעשה שימוש בערך ברירת מחדל, adOpenForwardOnly. בהמשך הפרק נדון בהשלכות של בחירה בערך מסוים, אך לעת עתה חשוב שתזכור שסמן "Forward Only" אינו תומך בשיטות הניווט MovePrevious ו-MoveLast.

שימוש בשיטות ניווט

לצורך הדגמת ניווט במערך רשומות ניצור כעת פרויקט שישתמש ב**שיטות ניווט** (Navigation Methods). התחל את הפרויקט ביצירת פרויקט חדש מסוג Standard EXE והוסף תיבת רשימה בשם lstData, שלושה לחצני פקודה בשם: cmdPrevious, cmdNext ו- cmdJump ותיבת טקסט שעבורה תגדיר את השם txtJump.

תוכנית הדוגמה תבצע שאילתה על מסד הנתונים BIBLIO ותאפשר למשתמש להפעיל שיטות ניווט לצורך עיון במערך הרשומות, באותו אופן בו עשית זאת כאשר השתמשת בפקד Data. דוגמת הקוד של פרויקט זה מוצגת בתוכנית 28.2.

תוכנית 28.2: ADONAV.VBP - שימוש בשיטות לניווט במסגרת מערך רשומות.

```
Option Explicit
Dim rs As ADODB.Recordset
Private Sub Form_Load()

    ' FILL THE RECORDSET OBJECT RS
    Set rs = New ADODB.Recordset
    rs.CursorType = adOpenStatic
    rs.Source = "Select * from Publishers"
    rs.Open, "DSN=BIBLIO"
    DisplayCurrentRecord

End Sub
Sub DisplayCurrentRecord()
    Dim i As Integer
    Dim s As String

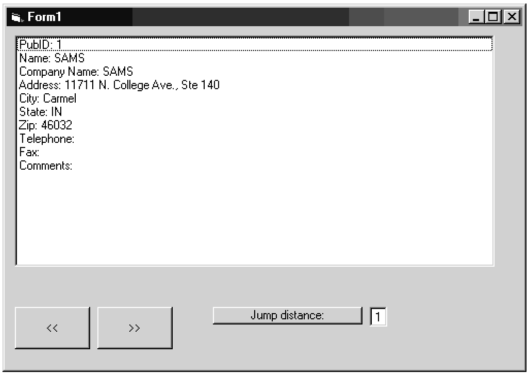
    If rs.BOF Then rs.MoveFirst
    If rs.EOF Then rs.MoveLast

    lstData.Clear
    For i = 0 To rs.Fields.Count - 1
        s = rs.Fields(i).Name & ": " & rs.Fields(i).value
        lstData.AddItem s
    Next i
End Sub
Private Sub cmdJump_Click()
    rs.Move Val(txtJump)
    DisplayCurrentRecord
End Sub

Private Sub cmdNext_Click()
    rs.MoveNext
    DisplayCurrentRecord
End Sub

Private Sub cmdPrevious_Click()
    rs.MovePrevious
    DisplayCurrentRecord
End Sub
```

אופן השימוש בשיטות ניווט בפרויקט זה ברור מאוד. החלק היחיד של התוכנית שדורש הסבר הוא הפונקציה DisplayCurrentRecord. שים לב שהפונקציה כוללת בדיקה לזיהוי מצבי BOF ו-EOF לפני שהיא מציגה את השמות והערכים של שדות הרשומה הנוכחית. בדיקה זו דרושה מפני שניסיון לגשת לאובייקט Field כאשר לא קיימת רשומה נוכחית יגרום לשגיאה. בתרשים 28.17 תוכל לראות תוכנית זו בפעולה.



תרשים 28.17: תוכנית הדוגמה תאפשר למשתמש לעיין במערך רשומות

עדכון נתונים

כעת, לאחר שאתה כבר יודע כיצד להכניס נתונים לאובייקט Recordset ולהציג אותם, תוכל לבצע את הצעד הבא ולעדכן (או לשנות) נתונים במסד הנתונים. אם הגדרת נכון את מערך הרשומות, תוכל לשנות נתונים במסד הנתונים בקלות. תידרש בסך הכל לנווט לרשומה המתאימה, להציב ערכים חדשים בשדות שאותם תרצה לשנות, ולקרוא לשיטה Update, באופן הבא:

```
rs.Fields("Author")="Simpson, Bart"
rs.Update
```

הערה:

כאשר השתמשת ב-DAO, נדרשת לקרוא לשיטה Edit **לפני שינוי** ערך שדה. ב-ADO **שינוי** ערך שדה **יעביר אוטומטית** למצב Edit, ב-ADO **לא מוגדרת** שיטת Edit. **כדי** לבדוק אם הרשומה הנוכחית נערכת כעת, בדוק את ערך **המאפיין** EditMode.

הגדרת מאפיין LockType

כברירת מחדל, מערכי רשומות ב-ADO הם לקריאה בלבד. ניסיון לשינוי ערך שדה במערך רשומות המוגדר לקריאה בלבד, יגרום לשגיאה. אם תרצה להוסיף, לגרוע, או לשנות רשומות, תידרש להציב במאפיין LockType של אובייקט Recordset ערך השונה מערך ברירת המחדל. לדוגמה, הגדרת המאפיינים CursorType ו-LockType באופן המודגם בקטע הקוד הבא תיצור מערך רשומות של Access שהוא בר-עדכון:

```
rs.CursorType = adOpenKeyset
rs.LockType = adLockOptimistic
```

אם תהית מהי חשיבות המאפיין LockType התשובה היא שכאשר תערוך רשומות במסד נתונים שיפעל בסביבה מרובת משתמשים, תידרש להתייחס לנושא **נעילת רשומות** (Record Locking). נעילת רשומות היא פעולה המונעת ממשתמשים שונים לבצע גישה לרשומה מסוימת, באותו זמן. המאפיין LockType משמש לשליטה על נעילת רשומות.

המאפיין יכול לקבל את הערכים המפורטים להלן :

- adLockReadOnly - מגדיר את מערך הרשומות כמערך לקריאה בלבד.
- adLockPessimistic - מיישם נעילת רשומות פסימית, כלומר נעילת רשומה בכל מצב שבו תבצע עליה פעולת עדכון כלשהי.
- adLockOptimistic - מיישם נעילת רשומות אופטימית, כלומר הרשומות ננעלות רק בעת קריאה לשיטה Update.
- adLockBatchOptimistic - מעדכן מספר רשומות בבת-אחת, תוך שימוש בשיטה UpdateBatch.

מדוע כונו הפרמטרים על שם גישות המיושמות באמצעותם (אופטימית ופסימית)? מפני שניסיון לבצע עדכון רשומה נעולה יגרום לשגיאה בתוכנית. תוכל להיות אופטימי ולהניח שרשומה תהיה זמינה לשימוש כאשר תרצה לעדכן אותה או להיות פסימי ולנעול את הרשומה כל עוד תזדקק לה.

הצגת שינויים שבוצעו על ידי משתמשים אחרים

אמינות נתונים היא שיקול נוסף שיש להתייחס אליו בעת עבודה עם מסדי נתונים בסביבות מרובות משתמשים. באחד הסעיפים הקודמים ציינו שמאפיין CursorType יכול להגביל את הניווט במערך הרשומות. אך הייעוד העיקרי של סוגי הסמנים המפורטים בטבלה 28.3 הוא שליטה על סוג החיבור שייושם בין מערך הרשומות לבין הנתונים שעליהם יתבסס.

טבלה 28.3: המאפיין CursorType

קבוע	תיאור
adOpenForwardOnly	הקבוע מאפשר פעולה מהירה, אך לנוע קדימה בלבד
adOpenKeySet	יאפשר לתוכנית לראות חלק מהשינויים שבוצעו על ידי משתמשים אחרים
adOpenDynamic	יאפשר לתוכנית לראות את כל השינויים שבוצעו על ידי משתמשים אחרים
adOpenStatic	יספק תמונה סטטית של מסד הנתונים, שלא תאפשר לראות שינויים שבוצעו על ידי משתמשים אחרים

בשימוש ב-ADO תגלה שלא כל מסדי נתונים תומכים בסוגים מסוימים של סמנים. במסדי נתונים של Access, סמנים מסוג adOpenForwardOnly מיועדים לבצע מעבר מהיר של קריאה בלבד, ואילו סמני adOpenKeySet מתאימים יותר לעדכון ופעולות מורכבות יותר.

הוספת נתונים חדשים

הוספת רשומות חדשות לאובייקט Recordset דומה לשינוי רשומות קיימות, אך דורשת פעולה אחת נוספת:

1. קרא לשיטה AddNew.
2. הצב ערכים בשדות.
3. קרא לשיטה Update.

הקוד שבתוכנית 28.3 יוסיף רשומה חדשה לטבלה Authors שבמסד הנתונים BIBLIO.

תוכנית 28.3: ADOTEST.VBP - הוספת רשומה למערך רשומות ב-ADO.

```
Dim cn As New ADODB.Connection
Dim rs As ADODB.Recordset

cn.Open "DSN=BIBLIO"

'OPEN A RECORDSET
Set rs = New ADODB.Recordset
rs.CursorType = adOpenKeyset
rs.LockType = adLockOptimistic
rs.Source = "Authors"
rs.ActiveConnection = cn
rs.Open

'ADD A NEW RECORD
rs.AddNew
rs.Fields("Author") = "King, Stevie"
rs.Fields("Year Born") = 1945
rs.Update
'NOTE: THE Au_ID field is an autonumber field,
'it will be created automatically.

'DESTROY THE OBJECTS
rs.Close
cn.Close
Set rs = Nothing
Set cn = Nothing
```

אובייקט Command

ברוב המקרים של עבודה עם סדרות נתונים, תעבוד עם מאפיינים ושיטות של אובייקט Recordset. אך עוד תגלה שאובייקט Command של ADO יועיל מאוד במצבים בהם תידרש לאחזר נתונים. אובייקט Command יאפשר לאחסן שאילתות SQL ופרוצדורות מאוחסנות של SQL באובייקטים הניתנים לשימוש חוזר, שיהיו מועילים במיוחד במצבים בהם תידרש לחזור על פעולות מספר פעמים. תוכל אפילו לאחסן פרמטרים של שאילתה או שיגרה באוסף Parameters של האובייקט, כך שלא תידרש להקדיש מחשבה לנושא בניית מחרוזות SQL עם קוד. לאחר שתגדיר אובייקט Command, תוכל לשנות את הפרמטרים שלו ולקרוא לו שוב ושוב.

לדוגמה, זכור לרגע את משפט SQL שהוצג בתוכנית 28.1, שבו נעזרנו לצורך בחירת סופרים ששם משפחתם מתחיל ב-Q. Access תאפשר לך להגדיר שאילתה גנרית בשם LetterLookup, תוך שימוש בתחביר המפורט להלן:

```
PARAMETERS Letter Text;
SELECT *
FROM Authors
WHERE Author Like Letter+'*';
```

אם תריץ שאילתה זו ב-Access, יוצג סמן שיורה לך להזין ערך עבור הפרמטר Letter ולאחר מכן תוצגנה הרשומות שתתאמנה לאות שתגדיר. אם תרצה להפעיל את השאילתה על ידי שימוש בקוד Visual Basic, תוכל להשתמש באובייקט Command, כמתואר להלן:

```
Dim cmd As New ADODB.Command
Dim parmTemp As ADODB.Parameter

Set cmd.ActiveConnection = cn
cmd.CommandText = "LetterLookup"
cmd.CommandType = adCmdStoredProc
Set parmTemp = cmd.CreateParameter("Letter", adChar, adParamInput, 1)
cmd.Parameters.Append parmTemp
cmd("Letter") = "Q"
Set rs = cmd.Execute
```

כדי שתוכל להשתמש באובייקט Command זה, תידרש לבחור את סוג הפקודה שתאוחסן בו (סוגי הפקודות פורטו בתחילת הפרק) ולהציב בו פקודה מסוימת. אם לאובייקט יידרשו פרמטרים כלשהם, תוכל לצרף אותם לאוסף Parameters של האובייקט. לאחר שתגדיר את אובייקט Command, תוכל פשוט לקרוא לשיטה Execute כדי לבצע את הפקודה.

מערכי רשומות מנותקים

אם עבדת בעבר עם טכנולוגיית DAO, תגלה שטכנולוגיית ADO מציעה יכולות נוספות המאפשרות לבצע פעולות נוספות, שבעבר לא יכולת לבצע, על מערכי רשומות. אחת היכולות היא ליצור **מערכי רשומות מנותקים** (Disconnected Recordsets). מערכי רשומות מנותקים הם מערכי רשומות המאוחסנים בזיכרון, שאינם תלויים בחיבור מסוים למסד נתונים. כאשר מערך רשומות אינו מחובר למסד נתונים, הוא מתפקד כאובייקט עצמאי, תוך שימוש בכל השיטות והמאפיינים הרגילים. אך בוודאי תופתע לגלות שלאחר שתבצע פעולות על מערך רשומות מנותק, תוכל לחבר אותו שוב למסד הנתונים, ולאחסן במסד הנתונים את השינויים שביצעת במערך הרשומות כאשר היה מנותק. בהמשך הסעיף יתברר לך שמערכי רשומות מנותקים מועילים במיוחד בעת עבודה בסביבת Web, מפני שבעת עבודה עמם אינך נדרש להיות מחובר כל הזמן למסד הנתונים. בסעיפים הבאים תלמד מהן הפעולות שתידרש לבצע לעבודה עם מערכי רשומות מנותקים.

יצירת מערך רשומות מנותק

הגדרת מאפיין CursorLocation של אובייקט Recordset היא המפתח ליצירת מערך רשומות מנותק. המאפיין יכול לקבל את הערכים: adUseClient ו-adUseServer. אם תרצה ליצור מערך רשומות מנותק (המכונה גם Client-side Cursor) יהיה עליך לבצע את הפעולות הבאות:

1. הגדר את המאפיין CursorLocation כ-adUseClient.
2. אחסן נתונים באובייקט Recordset.
3. הצב ערך Nothing במאפיין ActiveConnection של האובייקט.

קטע הקוד שהוצג בתוכנית 28.2 הציג את נתוני המוציאים לאור שבמסד הנתונים BIBLIO תוך שימוש בתיבת רשימה. תוכל לשנות את התוכנית כדי לגרום לה להשתמש במערך רשומות מנותק, על ידי הכנסת השינויים שלהלן בשגרת האירוע Form_Load:

```
Private Sub Form_Load()

        Set rs = New ADODB.Recordset
        rs.CursorLocation = adUseClient
        rs.CursorType = adOpenStatic
        rs.Source = "Select * from Publishers"
        rs.Open "DSN=BIBLIO"
        Set rs.ActiveConnection = Nothing
        DisplayCurrentRecord

End Sub
```

כדי שתוכל להשתמש במערכי רשומות מנותקים נדרשת להוסיף בסך הכל שתי שורות קוד לתוכנית שהוצגה בתוכנית 28.2. באחת הגדרת את המאפיין CursorLocation כדי לאחסן את הרשומות במחשב הלקוח, ולאחר מכן ביטלת את חיבור מערך הרשומות

למסד הנתונים על ידי כך שהרסת את אובייקט ActiveConnection שלו. אני מניח שבין הקוראים יהיו גם ספקנים שידרשו ראיות נוספות לכך שמערך הרשומות אכן נותק ממסד הנתונים. אם תרצה להוכיח זאת, הפעל את הפרויקט לאחר שתכניס בו את העדכונים הדרושים ליצירת מערך רשומות מנותק. כאשר הרשומה הראשונה תוצג על המסך, שנה באופן זמני את שם קובץ מסד הנתונים BIBLIO.MDB לשם אחר. לאחר שתבצע את השינוי חזור ל-Visual Basic ותגלה שפעולת התוכנית לא הושפעה מהשינוי. אך עם זאת לא תוכל להפעיל עותק נוסף של התוכנית, כל עוד קובץ מסד הנתונים BIBLIO.MDB לא יימצא.

חיבור מחדש של מערך רשומות

קודם לכן למדת כיצד לנעול רשומות כדי למנוע "מלחמות משתמשים" על שימוש ברשומה מסוימת. המושג נעילת רשומות מקבל משמעות חדשה בעת שימוש במערכי רשומות מנותקים. כאשר אתה לא מחובר למסד הנתונים, אתה למעשה "שואל" רשומות באותו אופן בו אתה שואל ספרים מספריה. אם תבצע שינויים ברשומות ששאלת, השינויים לא ישתקפו במערך הרשומות עד ש"תחזיר" את הרשומות למסד הנתונים, או אם להשתמש בטרמינולוגיה של מסדי נתונים עד שתבצע **עדכון אצווה** (Batch Update). כך שבעת עבודה עם מערכי רשומות מנותקים תידרש להשתמש באופן הנעילה Batch Optimistic.

פעולות עדכון אצווה מאפשרות לבצע כמה שינויים במערך הרשומות ולהחיל אותם על מסד הנתונים שעליו מבוסס מערך הרשומות, בבת-אחת, על ידי שימוש בשיטה UpdateBatch. תוכל גם לבצע את השינוי על מערך רשומות מנותק, להגדיר חיבור חדש בין מערך הרשומות למסד הנתונים, ולעדכן את מסד הנתונים.

כדי שתוכל להשתמש בשיטה UpdateBatch, תידרש לשנות את הגדרת המאפיין LockType של אובייקט Recordset לערך adLockBatchOptimistic :

```
rs.LockType = adBatchOptimistic
```

לאחר שתאחזר נתונים למערך הרשומות המנותק, כפי שנעשה בקטע הקוד שהוצג לעיל, תוכל לשנות רשומה או רשומות במערך על ידי הצבת ערכים חדשים בשדות :

```
rs.Fields(0) = "Edited Field Value"
rs.MoveNext
rs.Fields(0) = "Another Edited Field Value"
```

אחר, הגדר שוב את החיבור בין מערך הרשומות ומסד הנתונים, על ידי הצבת שם אובייקט Connection חדש במאפיין ActiveConnection של אובייקט Recordset :

```
Set cn = New ADODB.Connection
cn.ConnectionString = "DSN=BIBLIO"
cn.Open
Set rs.ActiveConnection = cn
```

לבסוף יהיה עליך להפעיל את קטע הקוד שיבצע את העדכון הממשי של מסד הנתונים:

```
rs.MarshallOptions = adMarshallModifiedOnly
rs.UpdateBatch
```

בוודאי שמת לב שבקטע הקוד שהוצג השתמשתי במאפיין חדש, MarshallOptions. מאפיין זה יגדיר האם פעולת העדכון תתבצע על ידי החזרת מערך הרשומות כולו, או רק של הרשומות ששונו, וזאת על בסיס שיקולי ניצול יעיל של משאבים. השורה השנייה מבצעת את העדכון, על ידי שימוש בשיטה UpdateBatch.

שימושים במערכי רשומות מנותקים

כעת, כאשר אתה כבר יודע מהם מערכי רשומות מנותקים, סביר להניח שאתה תוהה באילו נסיבות תרצה להשתמש במערכי רשומות כאלה. לא סביר להניח שבמסגרת עבודה בסביבת שרת/לקוח מקובלת, תידרש אי-פעם להשתמש בהם. אך מערכי רשומות מנותקים מאפשרים להוסיף עוצמה רבה ליישומים המיועדים לפעול בסביבת Web. תאר לעצמך מצב בו תוכל להריץ יישום Visual Basic שיפעל בדפדפן המתקשר עם שרת מסד נתונים שלך כפי שתוכל לעשות זאת באמצעות רשת תקשורת מקומית. כדי שתוכל לפתח יישומים רבי עוצמה כאלה, תידרש לפצל אותם לשני רכיבים:

- **לקוח Web** רכיב מסוג ActiveX OCX או מסמך ActiveX שיפעל בדפדפן.
- **שרת Web** רכיב מסוג ActiveX DLL שיפעל על גבי שרת Web.

בניית יישומים בעלי מבנה כזה תדרוש הבנת מוצרים מורכבים יחסית של Microsoft, שתיאוריהם חורגים מתחום עיסוק פרק זה. אך אם מובנים לך כמה מושגי יסוד רלוונטיים (כגון ADO ו-ActiveX Components), סביר להניח שתצליח במשימתך.

Microsoft משווקת מוצר ששמו Remote Data Services (RDS), המיועד לתפקד כחוליית קישור בין Internet Explorer ל-Internet Information Server. המפתח לשימוש ב-RDS הוא אובייקט DataSpace שלו. תוכל ליצור מופע של אובייקט RDS Data Space בלקוח Web, על ידי שימוש בשורת הקוד הבאה:

```
Set objRDS = CreateObject("RDS.DataSpace")
```

לאחר מכן תוכל להשתמש באובייקט Data Space ליצירת אובייקטים מרוחקים שיפעלו במסגרת שרת Web:

```
Set objRDS = CreateObject("MyDLL.MyClass", "http://server.somewhere.com")
```

שים לב להבדלים בין שתי שורות הקריאה לשיטה CreateObject. השורה הראשונה יוצרת אובייקט מקומי. השיטה השנייה יוצרת אובייקט מרוחק בשם objTest. תהליך יצירת האובייקטים דומה (אך אינו זהה) לשימוש בטכנולוגיית DCOM לשם יצירת אובייקט על מחשב מרוחק ופנייה לאובייקט זה לשם שימוש בגורמים הקשורים בו, תוך עבודה ממחשב מקומי.

אובייקט הדוגמה objTest הוא מופע של אובייקט מסוג ActiveX DLL שפועל על שרת Web. לאחר שתיצור מופע זה, תוכל לבצע קריאות לשיטות שלו, ולבצע פעולות להעברת נתונים אליו וממנו. ספריית האובייקטים ADOR, שהוזכרה בתחילת הפרק, היא אופן יישום מינימליסטי של מערך רשומות ADO (האות R שבשם הספריה היא קיצור של remote - מרוחק), המיועד לשימוש במסגרת פעולות כאלו. תוכל לכלול ברכיב ActiveX DLL שתיצור פונקציות שתשמשנה להעברת מערכי רשומות במבנה ADOR ללקוח Web, שיוכל להחזיר לשרת את הרשומות ששונו על ידי פעולת העיבוד.

מכאן....

פרק זה היווה מבוא לטכנולוגיית ActiveX Data Objects (ADO), המיועדת להחליף את הטכנולוגיות המיושנות יותר בתחום גישה לנתונים כדוגמת DAO ו-RDO. ADO מאפשרת לא רק לבצע פעולות סטנדרטיות של טיפול במסדי נתונים (כגון הוספה וגריעת רשומות) אלא גם להוסיף לתוכנית מיגוון יכולות רחב בתחום טיפול במסדי נתונים. בנוסף לממשק אובייקטים רב עוצמה, טכנולוגיית ADO מציעה גם מספר פקדים התומכים בה כגון DataGrid ו-ADO Data Control. מידע נוסף על נושאים שנדונו בפרק תוכל למצוא בפרקים הבאים:

- מידע נוסף על יישום החומר שבפרק זה במסגרת ה-Web תוכל למצוא בנספח 4 **דפי שרת פעילים**, ובפרק 32 **Visual Basic ושימושים נוספים באינטרנט**.
- מידע על הפקת דוחות על בסיס הנתונים תוכל למצוא בפרק 29 **יצירת דוחות**.
- מידע נוסף על חיבור בין פקדים ומסדי נתונים תוכל למצוא בפרק 25 **פקד הנתונים ופקדי איגוד נתונים**.

29

הפקת דוחות

מה בפרק?

- יצירת דוח לדוגמה
- העשרת דוחות נתונים
- Crystal Reports

אומנם משווקי מערכות מחשוב משרדיות מבטיחים שהמערכות שלהן תאפשרנה ניהול משרדי להתנהל ללא נייר, אך רבים עדיין מעדיפים להשתמש בדוחות מודפסים. לחלק ממפתחי Visual Basic הטיפול בדוחות הוא הקטע המתסכל ביותר בפיתוח יישומים. מפתחים המקדישים זמן רב בטיפול בדוחות וכותבים פונקציות משלהם תוך שימוש בשיטות Print ושיטות גרפיות אחרות, נוטים להקדיש פחות זמן לפיתוח חלקים אחרים של היישום. בקצה השני של הספקטרום נמצא השימוש בכלים להפקת דוחות, השימוש בהם הוא מהיר, אך הוא מעורר בעיות אחרות כגון עלויות גבוהות וצורך בהפצת קבצי DLL יחד עם היישומים.

ל- Visual Basic 6.0 נוסף כלי חדש שיאפשר לשלב דוחות ביישומים - Data Report. כלי זה מקל על הצגת מסכי תצוגה לפני הדפסה עם לחצני הדפסה וייצוא נתונים, ממקור נתונים המיישם את טכנולוגיית ADO. כל מה שנדרש הוא לספק את הנתונים ולהגדיר את מבנה הדוח. בנוסף לכלי זה, Visual Basic 6.0 תומכת גם בכלי Crystal Reports, שנכלל בחבילת Visual Basic מזה כמה גרסאות. כלי זה שפותח על ידי חברת Segate Software, מציע דרך קלה לעיצוב גרפי ולהפצת דוחות.

בפרק זה נסקור את שני הכלים ונלמד כיצד להשתמש בהם ביישומים.

יצירת דוח לדוגמה

הכלי Data Report הוא ActiveX Designer, כלומר אובייקט ActiveX מיוחד הניתן לשילוב בסביבת Visual Basic. בפרק 28 הוצג ISS (Internet Information Server) שגם הוא ActiveX Designer.

בסעיף זה תיצור דוח לדוגמה שיתבסס על שאילתה שתפנה למסד הנתונים BIBLIO. כדי ליצור את הדוח יש לבצע את הפעולות שלהלן:

1. הגדר אובייקט ADO Recordset שישמש את הדוח.
2. הוסף לפרויקט אובייקט DataReport.
3. עצב את הדוח על ידי מיקום שדות על טופס עיצוב אובייקט DataReport.
4. כתוב את הקוד שישמש להצגת הדוח.

הגדרת מקור הנתונים

התחל את פרויקט הדוגמה על ידי יצירת פרויקט חדש מסוג Standard EXE. הפעולה הבאה שתבצע אינה קשורה לדוח עצמו אלא לנתונים שיוצגו בו. לפני שתעצב דוח יש לדעת מהו המידע שעליו תדווח. כך שכעת תוכל להסתמך על הידע שרכשת בפרק 28 בנושא DAO, כדי ליצור שאילתה פשוטה שתהיה מאפיין DataSource של הדוח.

ראה, "הגדרת מקור נתונים", פרק 28

הגדר את השאילתה על ידי ביצוע הפעולות שלהלן, מידע על אופני הביצוע תוכל למצוא בפרק 28 :

1. הוסף לפרויקט הפניה לספריה Microsoft ActiveX Data Object 2.0 Library.
2. אם טרם הגדרת את מקור הנתונים BIBLIO ODBC, הגדר אותו כעת.
3. הצהר על משתני רמת הטופס cn ו-rs שייצגו את אובייקטי Connection ו- ADO Recordset.
4. שרטט לחצן פקודה על טופס היישום, הגדר את מאפיין Name כ-cmdFill ואת מאפיין Caption כ- Fill Recordset.
5. הוסף לשגרות האירוע Load ו-Unload של הטופס קוד שיפתח ויסגור את הקישור למסד הנתונים.
6. הוסף לשגרת אירוע Click של הלחצן את שורת הקוד הבאה שתאחסן במערך הרשומות את הרשומות שתוחזרנה כתוצאה מהפעלת שאילתת SQL הבאה :

```
"Select * from Titles where [Year Published] >= 1996"
```

תוכנית 29.1 תציג את כל הקוד שהוסף עד כה לתוכנית.

תוכנית 29.1: RPTDEMO.VBP - אחזור נתוני דוגמה של הדוח.

```
Dim cn As ADODB.Connection
Dim rs As ADODB.Recordset
Private Sub cmdFill_Click()

    Set rs = cn.Execute("Select * from Titles where [Year Published] >= 1996")

    MsgBox "Recordset Populated."

End Sub

Private Sub Form_Load()
    Set cn = New ADODB.Connection
    cn.Open "DSN=BIBLIO"
End Sub

Private Sub Form_Unload(Cancel As Integer)
    rs.Close
    cn.Close
End Sub
```

השאילתה שהצגנו תיצור מערך רשומות המכיל מספר רשומות מטבלת Titles שבמסד הנתונים BIBLIO.MDB. הטבלה כוללת מספר רשומות רב וכדי להאיץ את התהליך, הגבלנו את הבחירה לרשומות ספרים שיצאו לאור לאחר שנת 1996. כאשר תלחץ על לחצן הפקודה, אובייקט Recordset ששמו rs יאוכלס בנתונים שיתקבלו מטבלת Tables. אחר תוכל להוסיף אובייקטי Data Report, וללמוד כיצד להפיק דוח.

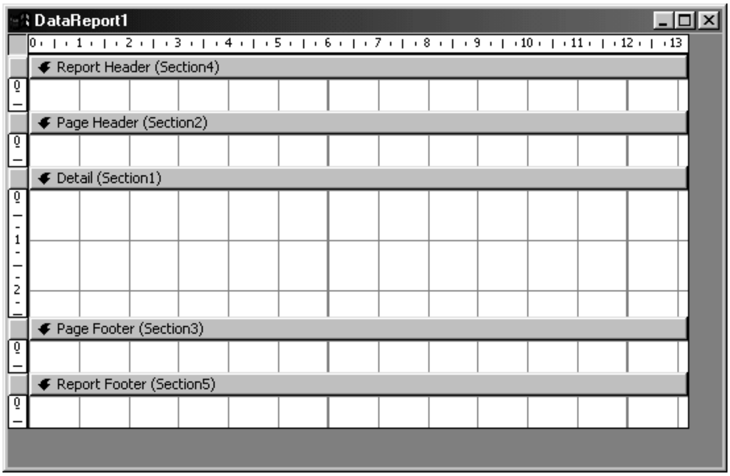
הוספת Data Report לפרויקט

כעת, כאשר כבר יש מקור נתונים, יש להגדיר כיצד הם יוצגו ב דוח. תחילה יש להוסיף אובייקט DataReport לפרויקט. כדי לעשות זאת, בחר **Add Data Report** מתפריט **Project**. לאחר שתעשה זאת, תבחין בכמה דברים שיתרחשו בבת-אחת:

- לחלון Data Explorer יתוסף אובייקט DataReport חדש בשם DataReport1.
- אזור DataReport חדש יתוסף לארגז הכלים שלך.
- יוצג חלון Report Designer, כמו זה הנראה בתרשים 29.1.

הערה:

כעיקרון, כל אובייקט DataReport מייצג דוח אחד. בהתאם לרמת המורכבות של הדוחות, ייתכן שתוכל להשתמש בקוד כדי לשייך מספר דוחות לאובייקט DataReport יחיד.



תרשים 29.1: הכלי Data Report מאפשר להגדיר את מבנה הדוח באופן חזותי

שים לב שהדוח מחולק למספר מקטעים, בדומה לדוחות של Access. מקטעים אלה משמשים להצגת חלקי הדוח המפורטים להלן:

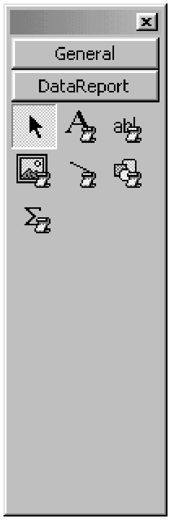
- **Report Header** (כותרת דוח). מידע המוצג פעם יחידה, בראש הדוח.
- **Page Header** (כותרת עליונה של עמוד). נתונים המוצגים בראש כל עמוד.
- **Detail Section** (מקטע פירוט). קטע הדוח שחוזר על עצמו עבור כל רשומה.
- **Page Footer** (כותרת תחתונה של עמוד). מידע המוצג בתחתית כל עמוד.
- **Report Footer** (כותרת תחתונה של דוח). מידע המוצג בסוף הדוח בלבד.

הערה:

תוכל גם להוסיף לכל דוח מקטעי Group Header ו- Group Footer, שיאפשרו לבצע להציג או לחשב שדות מסוימים במשך כל הדוח.

שים לב שלכל אחד מהמקטעים הוגדר שם מקטע (לדוגמה section1), המאפשר לגשת אליו מקוד התוכנית. אם תרצה להציג מאפייני מקטע מסוים, לחץ על שורת הכותרת האפורה של אותו מקטע. אם תרצה להציג את מאפייני אובייקט DataObject, לחץ על הריבוע המוצג בפינה השמאלית-עליונה של הטופס.

הגדרת דוח נתונים

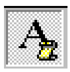
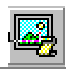
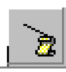
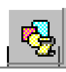
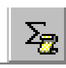


בעבודה במצב עיצוב (Design), תוכל להיעזר בחלון Data Report Designer כדי לסדר פקדים על הדוח, באותו אופן בו הוספת פקדים על טופס. סוגי השדות שתוכל למקם בדוחות הופכים זמינים לבחירה בארגז הכלים, כאשר אובייקט DataReport מצוי במוקד. שדות אלה מוצגים בתרשים 29.2.

תרשים 29.2: האלמנטים שתוכל להוסיף לדוחות נמצאים במקטע נפרד בארגז הכלים

אם תרצה להוסיף נתון לדוח, שרטט תחילה את הפקד המתאים במקטע הרצוי, והגדר את מאפייניו, כפי שתגדיר מאפייני כל פקד אחר. הפקדים שאפשר להוסיף לדוחות מוצגים בטבלה 29.1.

טבלה 29.1: פקדי Data Report

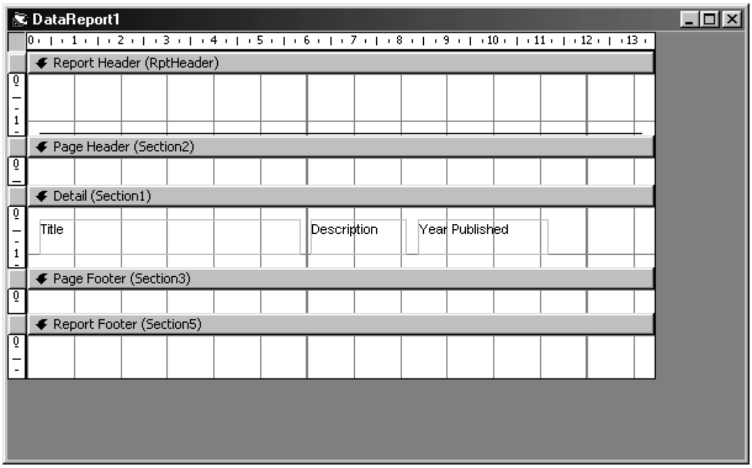
סמל	שם הפקד	תיאור
	RptLabel	מגדיר פקד Label שישמש להצגת טקסט שאינו מאוגד לנתון מסוים, כגון שם עמודה או תאריך הפקת הדוח
	RptTextBox	מציג תוכן שדה במסד הנתונים
	RptImage	מכיל תמונה או אובייקט גרפי אחר, כגון סמל חברה
	RptLine	מאפשר שרטוט קווים, למשל קווי הפרדה בין מקטעים
	RptShape	מאפשר שרטוט צורות לצורך הבלטה, או אפקטים חזותיים
	RptFunction	מאפשר לשלב בכותרת שדה פונקציה מתמטית, כגון סיכום

הוספת שדות למקטע פירוט

המשך ליצור את פרויקט הדוגמה ומקם מספר שדות בדוח. שרטט אובייקט RptTextBox במקטע פירוט של אובייקט DataReport1. שים לב שכאשר הפפקד יוצג, תוצג בו המילה Unbound, שתצביע על כך שהפקד טרם קושר לשדה במסד הנתונים.

הקש על F4 כדי להציג את חלון המאפיינים של תיבת הטקסט החדשה. שנה את שמה לשם בעל משמעות. לדוגמה, תוכל לקרוא לה txtTitle, מפני שהיא תשמש להצגת כותרי הספרים. אחר הצב את המילה Title במאפיין DataField של הפקד כדי לקשר את הפקד לשדה בטבלה. סגור את חלון המאפיינים ושים לב שהמילה titles מוצגת.

הוסף שני שדות טקסט נוספים למקטע פירוט, וקרא להם txtDesc ו-txtYear. הצב את הביטויים Description ו- Year Published במאפיין DataField של כל אחד מהפקדים, בהתאמה. לבסוף שנה את גודל המקטע על ידי ביטול השטח המבוזבז שמתחת לפקדים. הטופס שיווצר כתוצאה מתהליך זה מוצג בתרשים 29.3.



תרשים 29.3: הגדרת שדה בדוח פשוטה כמו הגדרת פקד איגוד נתונים

הוספת נתונים לכותרות

שלוש תיבות הטקסט שהוספת תוצגנה במקטע פירוט, כך שהן תשמשנה להצגת נתונים במסד הנתונים. אך יש להוסיף גם נתונים קבועים לכותרת הדוח, שיציינו מהם הנתונים המוצגים בתיבות הטקסט. עשה זאת על ידי הוספת פקדי RptLabel למקטעים המתאימים, והגדר את מאפיין Caption שלהם.

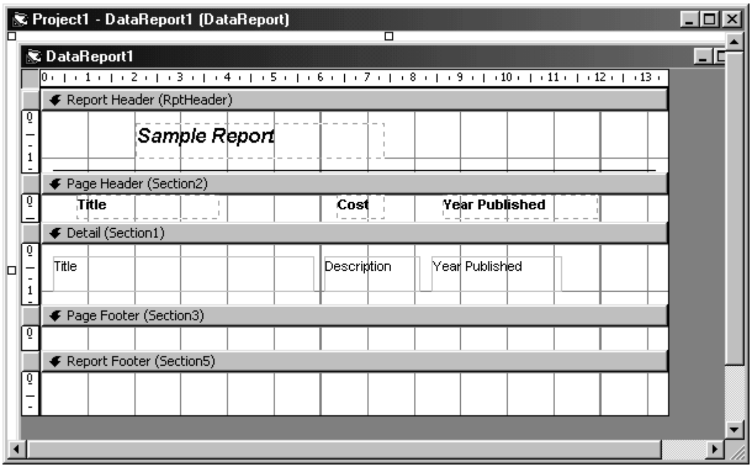
בדוח לדוגמה, הוסף שלושה פקדי תווית למקטע Page Header. הגדר את מאפיין Caption של כל אחד מהם Title, Cost ו- Year Published. סדר את פקדי התווית כך שכל אחד מהם יתייחס לשדה שיוצג מתחתיו במקטע פירוט Detail Section.

הוסף פקד RptLabel רביעי למקטע Report Header. בשדה זה יוצג שם האדם שיפיק את הדוח, שיוגדר באמצעות קוד. משום שיש לגשת לפקד התווית מקוד, כדאי שתגדיר למקטע ולפקד התווית שמות בעלי משמעות.

בצע את הפעולות שלהלן:

1. בחר את פקד RptLabel ושנה את הגדרת מאפיין Name לערך lblPerson, שנה גם את הגדרת מאפיין Caption למחרוזת ריקה.
2. לחץ על השורה האפורה שבראש מקטע Report Header והקש F4 כדי להציג את חלון המאפיינים של המקטע.
3. שנה את הגדרת מאפיין Name לערך rptHeader.

לאחר שתבצע את הפעולות, הדוח ייראה כמו בתרשים 29.4. בזאת סיימת את עיצובו.



תרשים 29.4: בתרשים מוצג מבנה דוח הדוגמה, לאחר הוספת השדות למקטע הכותרת

הצגת הדוח

כעת לאחר שהגדרת את מראה הדוח, הוסף מספר שורות קוד שתצגנה אותו. הוסף לטופס לחצן פקודה והגדר את מאפיין Name ל- cmdReport ואת מאפיין Caption ל- Display Report. הוסף את קטע הקוד הבא לשגרת אירוע Click של הלחצן.

תוכנית 29.2: RPTDEMO - הצגת הדוח בפני המשתמשים.

```
Private Sub cmdReport_Click()
    Dim s As String
    s = "Prepared by Abe Froman"

    Set DataReport1.DataSource = rs
    DataReport1.Sections("rptHeader").Controls("lblPerson").Caption = s
    DataReport1.Show

End Sub
```

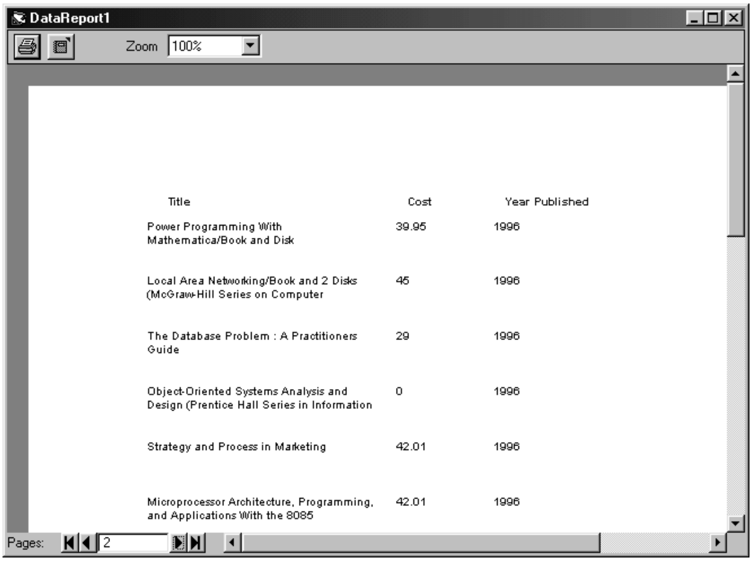
קטע הקוד המוצג בתוכנית 29.2 מציב את שם אובייקט ADO Recordset, rs, במאפיין DataSource של אובייקט DataReport.

הערה:

בדוגמה שהובאה זה עתה השתמשת באובייקט ADO Recordset כבמקור נתונים. אך ניתן להשתמש גם במקורות נתונים מסוגים אחרים, כגון Data Environment, שמאפשרים להגדיר את המאפיין DataSource במצב עיצוב.

לאחר מכן, יוגדר מאפיין Caption של פקד התווית שבכותרת הדוח. לבסוף תתבצע קריאה לשיטה Show, שתגרום להצגת הדוח.

אם תרצה לנסות את הדוח לדוגמה, הפעל את הפרויקט ולחץ על לחצן Fill Recordset. לאחר שתוצג תיבת הודעה שתציין שמערך הרשומות אוכלס בנתונים, לחץ על לחצן Display Report. הדוח יוצג בחלון נפרד, בתצוגה לפני הדפסה, כמתואר בתרשים 29.5.



תרשים 29.5: כך ייראה דוח הדוגמה לאחר השלמתו

העשרת הדוחות

בסעיף הקודם למדת כיצד תוכל להפיק דוחות בקלות, על ידי שימוש בכלי Report Designer. הדוח לדוגמה שנוצר היה אומנם גולמי מאוד, אך ענה על צרכיך. סביר להניח שבעת בניית הדוח, הבחנת שתוכל לבצע שינויים רבים בשדות שבדוח, למשל במאפיין Font, שאינו דורש הסבר נוסף. כדי לעצב את הדוח באופן הרצוי לך, יש בסך הכל להפעיל את הכשרונות האומנותיים. בכל מקרה, חשוב שתזכור שהכלי Data Report מציע מיגוון יכולות נרחב, שחלקן תתוארנה בסעיפים הבאים.

שדות דוחות מוגדרים מראש

במהלך הדוגמה ראית שאובייקט DataReport מאפשר לפנות למאפייני הפקדים על ידי שימוש בקוד. לדוגמה, אם תרצה להוסיף לדוח שדה שיציג את תאריך יצירתו, תוכל להגדיר את מאפיין Caption של פקד התווית Now בטרם תציג את הדוח. תוכל להשיג את אותה תוצאה גם בדרך פשוטה יותר. אובייקט Data Report תומך בכמה סוגים מוגדרים מראש של **מצייני מיקום** (Place Holders), שמאפשרים לשלב בדוחות כמה אלמנטים דינמיים. מצייני המיקום יפורטו בטבלה 29.2.

הערה:

המאפיין Title משמש גם לזיהוי הדוח לתיבות הדו-שיח של מדפסות.

טבלה 29.2: מצייני מיקום מוגדרים מראש בדוחות

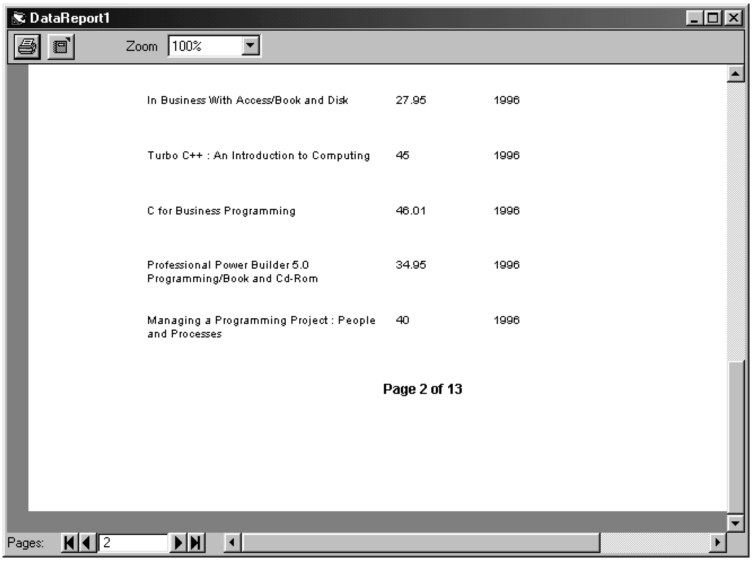
קוד	תיאור
%p	מספר העמוד הפעיל
%P	מספר עמודים כולל
%d	תאריך עדכני (מוצג במבנה קצר), לדוגמה, 5/24/98
%D	תאריך עדכני (מוצג במבנה ארוך), לדוגמה, Sunday, May 24, 1998
%t	שעה עדכנית (מוצגת במבנה קצר), לדוגמה, 16:10
%T	שעה עדכנית (מוצגת במבנה ארוך), לדוגמה 4:10:00 PM
%i	כותרת דוח (המחרוזת המאוחסנת במאפיין Title של אובייקט DataReport)

לשימוש באחד ממצייני מיקום אלה, הוסף לטופס פקד RptLabel והצב במאפיין Caption שלו ביטוי שישלב את מציין המיקום הרצוי. לדוגמה, אם תרצה להציג מספר עמוד בתחתית העמוד, הוסף פקד תווית לקטע הכותרת התחתונה של העמוד, והצב במאפיין Caption שלו את המחרוזת הבאה:

```
Page %p of %P
```

הרץ את הדוח ו- Data Report יחליף את מצייני המיקום בנתונים המתאימים, כמתואר בתרשים 29.6.

אם תרצה להשתמש באחד ממצייני המיקום בפני עצמו, מבלי להוסיף טקסט משלך, תוכל להוסיף אותו לדוח בדרך מהירה. כל שיש לעשות לשם כך הוא ללחוץ לחיצה ימנית על הדוח ולבחור מתוך התפריט שיוצג את הפקודה **Insert Control**. תוכל להשתמש בשיטה זו לכל אחד מהשדות שהוגדרו מראש ולכל פקד אחר המוצע לשימוש.



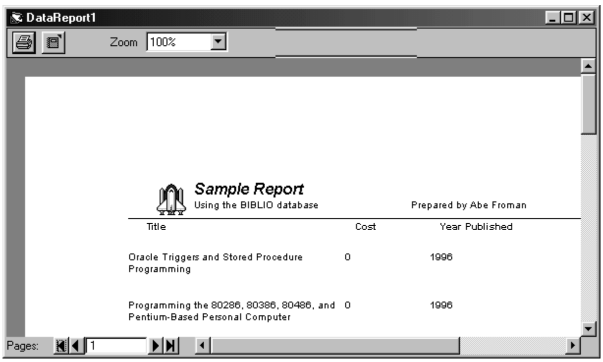
תרשים 29.6: הוספנו מונה עמודים לדוח הדוגמה

הוספת אלמנטים גרפיים

צורות גרפיות וקווים אומנם אינם מוסיפים לדוחות יכולות פונקציונליות כלשהן, אך הם משפרים את חזות הדוחות. כך שתוכל להשתמש באלמנטים כאלה לשם הבלטת דברים, או לסידור חזותי של הנתונים. שלושה פקדים מאפשרים להוסיף אלמנטים גרפיים לדוחות: RptImage, RptLine ו-RptShape.

תמונות

כדי להוסיף תמונה לדוח, יש לשרטט תחילה פקד RptImage בחלון עיצוב הדוח. אחר, טען תמונה לתוך הפקד, על ידי הצגת חלון המאפיינים של הפקד ולחיצה על הלחצן בונה (שלוש הנקודות) בשורת המאפיין Picture. בחר קובץ תמונה. תרשים 29.7 יציג סמל חברה שהוסף לכותרת הדוח באמצעות שימוש בפקד RptImage.



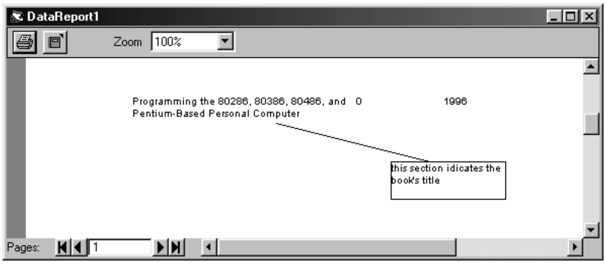
תרשים 29.7: ניתן להוסיף תמונות לדוחות על ידי שימוש בפקד RptImage

פקד RptImage אומנם דומה לפקד תמונה רגיל, אך חשוב לזכור שהוא פקד נפרד שהוגדרו עבורו מאפיינים משלו. להלן כמה הבדלים החשובים בין שני פקדים אלה:

- מאפיין PictureAlignment מאפשר להגדיר את מיקום התמונה בפקד, באופן דומה להגדרת יישור טקסט בפקד תווית.
- פקד - RptImage מציע מיגוון אפשרויות רחב יותר להגדרת מאפיין BorderStyle מכפי שמציע פקד Image רגיל.
- מאפיין SizeMode של פקד RptImage רב-תכליתי יותר ממאפיין Stretch של פקד Image. פקד RptImage תומך גם בהגדרת ערך Zoom המאפשר להגדיל את התמונה עד שהיא חורגת מגבולות הפקד.

קווים וצורות

פקד RptImage מאפשר לשלב תמונות בדוחות ואילו פקדי RptLine ו-RptShape מאפשרים לשלב קווים וצורות גיאומטריות אחרות. אופני הפעולה של פקדים אלה דומים לאלה של פקדי Line ו-Shape שתוארו בפרק 19. בתרשים 29.8 מוצג דוח שהועשר בקווים ובמלבנים לשם הבלטת הנתונים.



תרשים 29.8: שילוב צורות במקטע פירוט מאפשר להציג אותן שוב ושוב בכל רשומה

הדפסה וייצוא

במהלך הפרק נוכחת שתהליך עיצוב הדוח מורכב מהגדרת מקור נתונים שעליו הוא יתבסס, ומהגדרת חזות/פריסת הדוח. לאחר שתפיק דוח מסוים תעמודנה לרשותך שלוש דרכים שתאפשרנה להעביר אותו למשתמשים:

- קריאה לשיטה Show כדי להציג את הדוח בחלון תצוגה מקדימה. תצוגה כזו מאפשרת למשתמשים לדפדף בדוח ולהדפיס עותק שלו.
- להשתמש בשיטה PrintReport כדי לעקוף את חלון התצוגה המקדימה ולהדפיס את הדוח ישירות למדפסת. בשיטה זו כדאי להשתמש בפעולות להפקת דוחות ממוכנת או ביישום שמציג את הנתונים, ולכן אין הכרח לבצע תצוגה מקדימה.
- שימוש בשיטה ExportReport לשם שמירת תוכן הדוח בקובץ. שיטה זו יכולה לשמש ליצירת דפי Web מדוחות.

באחת הדוגמאות הקודמות הצגנו שימוש בשיטה Show לצורך הצגת דוח. השימוש בשתי השיטות האחרות הוא פשוט באותה מידה.

שימוש בשיטה PrintReport

להלן שורת קוד המיישמת את אופן הקריאה הפשוט ביותר לשיטה PrintReport :

```
DataReport1.PrintReport
```

השיטה מציעה גם מספר פרמטרים אופציונליים, שהשימוש בהם יקנה רמת שליטה גבוהה יותר על פעולת ההדפסה :

- ShowDialog - פרמטר הקובע האם תוצג תיבת הדו-שיח **הדפסה** של Windows, שתאפשר למשתמש לבטל את ההדפסה או להגדיר פרמטרים (ערך ברירת המחדל של פרמטר זה הוא False).
- Range - פרמטר המגדיר האם יודפסו כל עמודי הדוח, או רק עמודים נבחרים. ערך ברירת המחדל הוא rptRangeAllPages, שגורם להדפסת כל העמודים.
- PageFrom ו- PageTo - אם בפרמטר Range מוצב הערך rptRangeFromTo, שני הפרמטרים האלה יגדירו את העמוד הראשון והעמוד האחרון של ההדפסה.

בשורת הקוד הבאה נכללו כל הפרמטרים כדי להדפיס את עמוד 5 בלבד :

```
DataReport1.PrintReport True, rptRangeFromTo,5,5
```

הצבת ערך True בפרמטר ShowDialog תאפשר למשתמשים לשנות את הפרמטרים של פעולת ההדפסה ואף לבטלה.

שימוש בשיטה ExportReport

ייצוא דוח לקובץ HTML או לקובץ טקסט מתבצע באמצעות השיטה ExportReport. לדוגמה, שורת הקוד הבאה תיצור דף HTML שיכיל בתוכו את הדוח :

```
DataReport1.ExportReport "key_def_HTML", "D:\Temp\MyReport.HTM", True, False
```

בדומה לשיטה PrintReport, גם לשיטה זו יש כמה פרמטרים אופציונליים :

- FormatIndexOrKey - מגדיר את פורמט קובץ הייצוא. לאוסף ExportFormats ארבעה סוגים אפשריים, למעשה יש לבחור בין קובץ HTML לבין קובץ טקסט.
- FileName - מציין את הנתיב המפורט לקובץ שאותו אתה מעוניין ליצור.
- Overwrite - מגדיר פרמטר בוליאני (שערך ברירת המחדל שלו הוא True) המציין האם לדרוס את קובץ הייצוא. אם מוצב ערך False, הניסיון יגרום לשגיאה.
- ShowDialog - קובע האם תיבת הדו-שיח Export תוצג לפני המשתמש.
- Range, PageFrom, PageTo - משמשים להגדרת העמודים שייוצאו. אופני הפעולה זהים לאלה של הפרמטרים המקבילים בשיטה PrintReport.

חשוב שתדע שבנוסף לכך שמתאפשר למשתמשים להשתמש בקוד Visual Basic לשם הדפסה וייצוא הדוחות, הם יכולים לבצע פעולות אלו גם באופן ידני, תוך שימוש בלחצנים המוצגים בחלק העליון של חלון התצוגה המקדימה.

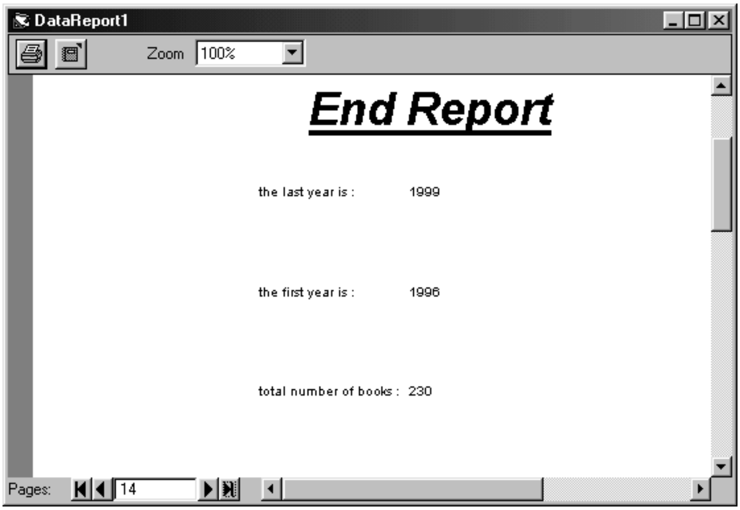
שדות מבוססי פונקציות

כבר למדת כיצד להוסיף שדות לדוחות לצורך הצגת מידע סטטי או נתונים המאוחסנים במסד הנתונים. אך דוחות יכולים להכיל גם שדות שהערכים שיוצגו בהם יחושבו באמצעות הפעלת פונקציות פשוטות. שדות כאלה מכונים שדות פונקציות (Function Fields) והוספתם לדוח מתבצעת באמצעות פקד rptFunction.

הערך שיוצג בשדה פונקציה תלוי בהגדרות שני מאפיינים, מאפיין DataField מגדיר את השדה שעליו תופעל הפונקציה, והמאפיין FunctionType מגדיר מהי הפונקציה שתופעל. במאפיין זה ניתן להציב את הערכים הבאים :

- rptFuncSum - סכום.
- rptFuncAve - ממוצע.
- rptFuncMin - ערך מינימום.
- rptFuncMax - ערך מקסימלי.
- rptFuncRCnt - ספירת שורות.
- rptFuncVCnt - ספירת ערכים, מספר השורות המכילות ערך מסוים.
- rptFuncSDEV - סטיית תקן.
- rptFuncSERR - שגיאת תקן.

פקד הפונקציות יכול לשמש להפעלת אחת הפונקציות על שורות רבות באחת העמודות של מערך הרשומות והצגת התוצאה. פקדי פונקציה מוצבים במקטע הכותרת התחתונה, מפני שיש לעבד את כל הנתונים הרלוונטיים לפני חישוב את הערך שיוצג.



תרשים 29.9: ניתן להיעזר בפקדי פונקציות להצגת סכומים, ממוצעים ומספרי רשומות

הערה:

ניתן גם לשלב פקדי פונקציות ב- Group Footers, כדי לאפשר הצגת תוצאות ביניים. מידע על שימוש במקטעים Group Headers ו- Group Footers תמצא בעזרה המקוונת של Visual Basic.

כדי להציג דוגמת שימוש בפקדים להצגת ערכים מבוססי פונקציות, נתייחס לטבלה פשוטה הכוללת שתי עמודות SalesPerson ו- Revenue. בתרשים 29.9 מוצג דוח הנעזר בפקדי פונקציה כדי להציג את סך הפדיון ואת הפדיון הממוצע לאיש מכירות.

הפונקציות Average ו-Sum הופעלו על שדה Revenue לחישוב הערכים המוצגים בשורה התחתונה של הדוח המוצג בתרשים.

Crystal Reports

Crystal Reports הוא כלי להפקת דוחות מתוצרת יצרן חיצוני שמשווק ב- Visual Basic. כלי זה מאפשר להפיק דוחות המבוססים על סוגי מקורות נתונים רבים. תוכל לתכנן דוחות באופן גרפי, ולנסותם באופן גרפי תוך שימוש בכלי העיצוב של Crystal Reports (Crystal Reports Designer). נתוני עיצוב הדוח ונתונים נוספים הקשורים בו מאוחסנים בקובץ בעל מבנה ייחודי בעל סיומת RPT. לאחר שתיצור קובץ ב- Crystal Reports תוכל להציג את הדוח לפני המשתמש באחד האופנים הבאים :

- פקד Crystal Reports מאפשר לשלב דוחות מוגמרים בתוכניות Visual Basic בצורה פשוטה.
- קריאות API של Crystal Reports מאפשרות לפנות למנגנון Crystal Reports מבלי להשתמש בפקד מותאם. כל הקריאות API של Crystal Reports ממוקמות בקובץ GLOBAL32.BAS שאמור להימצא בתיקיה בה נמצא קובץ ההפעלה של Crystal Reports.
- Crystal Report Web Server, שבמהדורת Crystal Reports Proffessional פועל בשרת Web ומאפשר להעביר דוחות ליישומי לקוח הפועלים בסביבת Web.

הערה:

נכון לכתיבת דברים אלה, הגירסה העדכנית ביותר של Crystal Reports היא גירסה 6.0. אך Visual Basic תומכת בגירסה ישנה יותר (גירסה 4.6). כדי שתוכל להשתמש בגירסה העדכנית, יש להתקין אותה. התחל את התקנת Crystal Reports על ידי לחיצה כפולה על קובץ ההפעלה Crystal32.exe שאותו תמצא בתיקיה \Common\Tools\Crysrept בתקליטור מספר 3 של חבילת ההתקנה Visual Studio (Crystal Report מצורף לחבילת Visual Studio ולא לתוכנת Visual Basic עצמה).

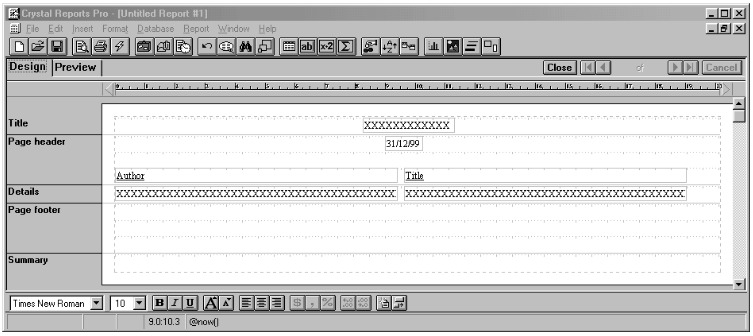
הסעיפים הבאים יציגו את הפעולות הבסיסיות של Crystal Reports ויתארו בפירוט את הפעולות הדרושות להצגת דוח תוך שימוש בפקד Crystal Reports.

יצירת דוח חדש

בהנחה ש- Crystal Reports הותקן בהצלחה, תוכל להפעיל את כלי העיצוב (Crystal Reports Designer) מתפריט **Add-ins** של Visual Basic. Report Designer, שמוצג בתרשים 29.10, הוא סביבת פיתוח לקבצי דוחות. סביבת הפיתוח מאפשרת לעצב דוחות חדשים, לבצע שינויים בדוחות קיימים ולהציג תצוגה מקדימה דוחות מוכנים.

הערה:

CRW32.exe הוא קובץ הפעלה המשמש להפעלת Crystal Reports Designer. בעת שנכתבו שורות אלו, תוכנית ההתקנה של Visual Basic לא יצרה סמל עבור Crystal Reports Designer, מצב הגורם לכך שהמשתמשים נדרשים לאתר את קובץ ההפעלה בדיסק שלהם, ולהגדיר קיצור דרך אליו, כדי שאפשר יהיה להפעילו גם מחוץ לסביבת Visual Basic (מיקום ברירת המחדל של הקובץ הוא Program Files\MS Visual Studio\Common\Tools\Reports). אם 6.0 Crystal Reports מותקנת, סביר להניח שתוכנית ההתקנה יצרה את קבוצת יישומי Crystal reports בתפריט ההתחלה והוסיפה סמל קיצור דרך.

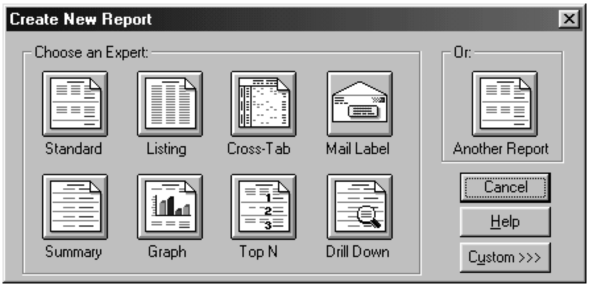


תרשים 29.10: Crystal reports Designer הוא יישום נפרד מ- Visual Basic.

בתרשים 29.10 תבחין בשתי כרטיסיות המגדירות את מצבי תצוגת הדוח: Design ו- Preview. בעת עבודה במצב Design, תוכל לשרטט ולסדר שדות על הטופס. התהליך דומה לזה המיושם בעת שרטוט פקדים על טפסים. יש לשרטט את השדה, להגדיר שם וכמה מאפיינים נוספים. אם תעבור לכרטיסיית Preview, Crystal Reports תתקשר למקור הנתונים ותציג נתונים בדוח.

ננסה את Crystal Reports על ידי הגדרת דוח לדוגמה, שיתבסס על מסד הנתונים BIBLIO. פתח את Crystal Reports Designer והתחל ביצירת הדוח, בחר **New** מתפריט **File**. תוצג תיבת הדו-שיח **Create New Report**, המוצגת בתרשים 29.11.

תיבת הדו-שיח תאפשר לבחור מבין כמה סוגי דוחות. אם תבחר את אחד, תוכל להיעזר ב- Create Report Expert. המומחה מחלק את תהליך היצירה לכמה שלבים, בדומה לאופן הפעולה של אשף. תוכל גם ליצור דוח מאפס, על ידי לחיצה על לחצן Custom ובחירת שתי אפשרויות מהחלק שיופיע: סוג הדוח ומקור נתונים.

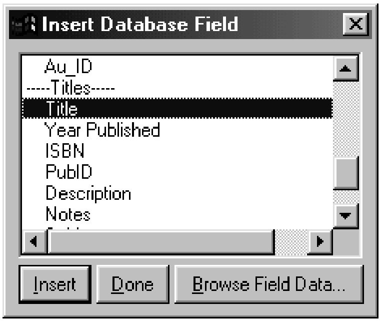


תרשים 29.11: Crystal Reports מאפשר לעצב דוח מאפס או להסתייע במומחה. לאחזור הנתונים תוכל להסתייע ב-ODBC או באחת ממנהלי ההתקן המוצעים

יצירת דוח מאפס

נמשיך את יצירת דוח הדוגמה שיתבסס על מסד הנתונים BIBLIO. לחץ על הלחצן Custom כדי להציג תיבת דו-שיח לבחירת סוג הדוח וסוג הנתונים. ודא שלחצן Custom Report נבחר ולחץ על לחצן Data File. כעת יש להגדיר את מיקום קובץ מסד הנתונים BIBLIO.MDB (בדרך כלל מאוחסן בתיקיה של Visual Basic).

בהנחה שהמערכת שלך תקינה, לאחר שתבחר את קובץ מסד הנתונים שבו תשתמש, תוצג תיבת הדו-שיח Insert Database Field אשר מוצגת בתרשים 29.12. תיבת הדו-שיח תוצג באופן אוטומטי, אך תוכל גם להציג אותה בעזרת תפריט **Insert**. תיבת דו-שיח זו תאפשר לבחור את השדות שבדוח.



תרשים 29.12: בחר פריטים מרובים מתיבת הדו-שיח Insert Database Field על ידי החזקת מקש Ctrl לחוץ ולחיצה על הפריטים הרצויים

לצורך דוח הדוגמה, בחר את השדה Author מטבלת Authors ואת השדה Title מטבלת Titles (מסד הנתונים BIBLIO מכיל את הקישורים הדרושים). אם יש צורך, תוכל לשנות את מיקום תיבת הדו-שיח כדי שהיא לא תסתיר את החלון הראשי של כלי העיצוב. לאחר מכן גרור את השדות שבחרת מתיבת הדו-שיח למקטע Details של הדוח. סדר את השדות כך שהדוח ייראה כמו זה שמוצג בתרשים 29.13. לחץ על לחצן Done כדי לסגור את תיבת הדו-שיח Insert Database Field.

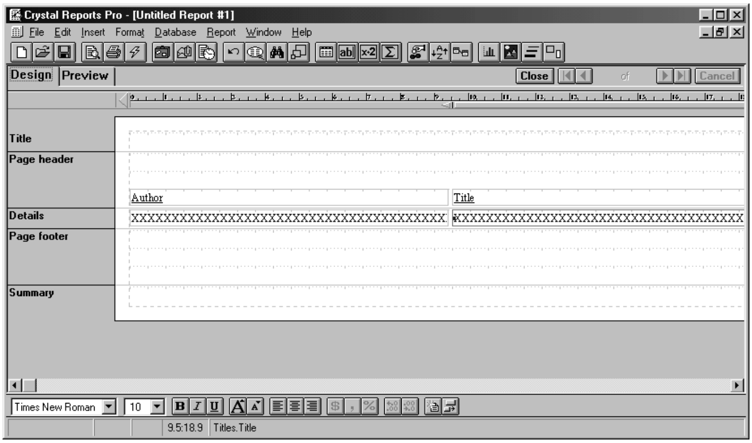
לבסוף, תוכל לנסות את הדוח על ידי לחיצה על הלחצן בעל סמל הברק שבסרגל הכלים. לחיצה על לחצן זה מפעילה את התצוגה המקדימה, שמאפשרת לראות איך ייראה הדוח בעת עבודה עם רשומות הלקוחות ממסד הנתונים. שמור את הדוח על ידי לחיצה על לחצן Save, או על ידי בחירה בפקודה **Save** מתפריט **File**.

טיפ:

אם אתה מתכוון להשתמש בתכנית להצגת הדוח עם נתונים שונים בכל פעם, כדאי שתנטרל את האפשרות Save Data with Report. לאפשרות זו תוכל להגיע על ידי בחירה בפקודה **Options** מתפריט **File** ובחירה בכרטיסיה **Reporting**.

יצירת דוח בעזרת Create Report Expert

דרך נוספת ליצירת דוחות היא על ידי שימוש ב-Create Report Expert. בחר את סוג הדוח הרצוי מתיבת הדו-שיח Create New Report. להלן סוגי הדוחות שיוצעו :



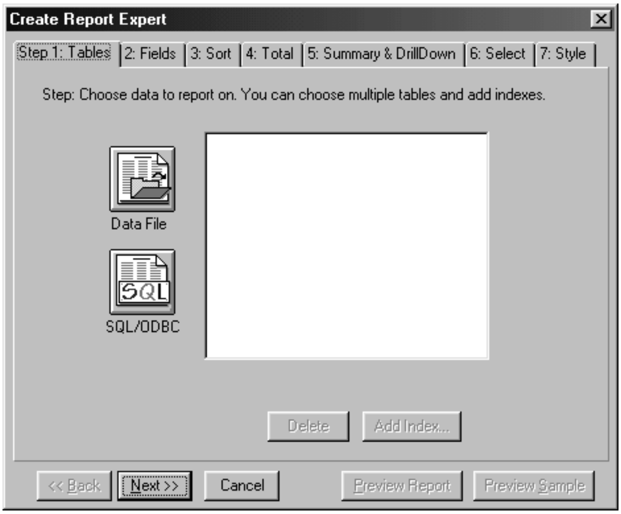
תרשים 29.13: כאשר תגרור שדה מסד הנתונים למקטע Details, Crystal Reports יוסיף התאמות למקטע Header של הדוח, באופן אוטומטי

- **Standard** (רגיל). מבנה דוח סטנדרטי הכולל שורות ועמודות. דוחות מסוג זה מכילים במקרים רבים נתוני סיכום בתחתית העמודות. הם גם מאפשרים לקבץ נתונים תוך שימוש בקריטריונים מוגדרים.
- **Listing** (רשימה/פירוט). דוח סטנדרטי שהנתונים המוצגים בו מוצגים במבנה רשימה. זהו המבנה המתאים להצגת דוחות כגון רשימות עובדים או לקוחות.
- **Cross Tab** (דוח הצלבות). דוח זה מציג דוחות בסדר הפוך מזה המקובל בדוחות רגילים. עמודות הדוח מייצגות את הרשומות במערך הרשומות. דוחות כאלה משמשים להצגת סיכומים המתבססים על מערכי נתונים מורכבים יותר.
- **Mail Label** (תוויות דואר). מבנה דוח כזה משמש להפקת תוויות דואר בהתבסס על נתונים שבמסד הנתונים.

- **Summary** (סיכום/תמצית). דוח כזה מציג את סיכום הנתונים מבלי להציג את הנתונים עצמם. ברוב המקרים הנתונים מקובצים לקבוצות ולכל קבוצה מחושבים נתוני סיכום.
- **Graph** (תרשים)**.** בחירה באפשרות זו תעביר דרך סדרת פעולות שתסייענה בהגדרת התרשים.
- **Top N** (N רשומות עליונות). דוח כזה יתייחס רק למספר מוגדר של רשומות מתוך מערך רשומות. הוא יכול לשמש למשל לשם הצגת נתוני המכירות של חמשת אנשי המכירות המובילים בארגון.
- **Drill Down** (דוח "קידוח"). דוח כזה מציג נתוני עזר או נתוני פירוט עבור כל רשומה ורשומה.

תרשים 29.14 מציג את מסך הפתיחה של Create Report Expert, שמאפשר להגדיר את מקור הנתונים שעליו יתבסס הדוח. לאחר שתשלים את הפעולה, יש להגדיר פרטים נוספים הדרושים לשם השלמת הדוח.

כאשר תסיים ליצור את הדוח תוכל להוסיף ולגרוע פרטים, כפי שעשית בעבודה עם דוחות מותאמים אישית.



תרשים 29.14: ל- Create Report Expert כמה כרטיסיות, שכל אחת מהן מייצגת שלב בתהליך יצירת הדוח

התאמה אישית של הדוח

גם אם נעזרת ב-Create Report Expert להגדרה ראשונית של מבנה הדוח, סביר להניח שתהיה מעוניין להכניס שינויים. למרבה המזל, Crystal Reports מציע מיגוון רחב של אפשרויות להתאמה אישית. תיאור מפורט חורג מתחום עיסוק ספר זה, אך בפרק זה נעסוק בשני סוגים חשובים להתאמה אישית, הוספת שדות חדשים והגדרת קריטריונים לבחירת רשומות.

סוגי שדות

כבר ציינו שפריסת שדות בדוח דומה לפריסת פקדים על טופס. תפריט **Insert** מאפשר לבחור מבין כמה סוגי שדות:

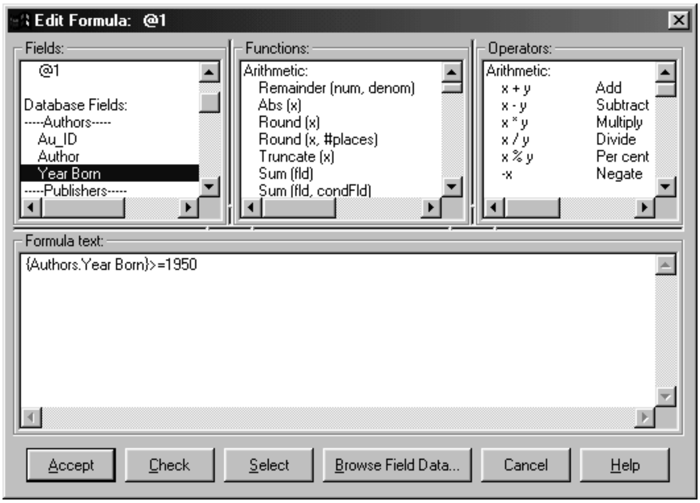
- **Database Field** (שדה מסד נתונים). מקשר בין פריטים בדוח לבין שדות במקור הנתונים. אופן הפעולה דומה לזה של פקדי איגוד נתונים.
- **Text Object** (אובייקט טקסט). מספק מידע טקסטואלי קבוע, כגון כותרות.
- **Formula Field** (שדה נוסחה). משמש להצגת תוצאות נוסחאות (ביטויים). הביטויים בשדות כאלה מתבססים על שדה או שדות במסד הנתונים.
- **Parameter Field** (שדה פרמטר). מאפשר להגדיר פרמטרים שניתן להעביר לדוח, באמצעות תוכנית או באופן ידני. שדות פרמטרים ניתנים לשימוש בקריטריונים לבחירת רשומות ובנוסחאות אחרות.
- **Special Field** (שדה מיוחד). מציג באופן אוטומטי נתונים כגון מספר העמוד הפעיל, השעה העדכנית או נתונים מועילים אחרים.

כל מה שעליך לעשות כדי להוסיף שדה לדוח שלך הוא לבחור אותו מהתפריט, ולהגדיר את הנתונים הדרושים לשם שימוש בו. אם תרצה לגרוע שדה קיים מקובץ, בחר אותו והקש Delete. תוכל גם ללחוץ לחיצה ימנית על שדה, כדי להציג תפריט מקוצר שיאפשר לשלוט במאפייני השדה, כגון תוכן ועיצוב.

הוספת נוסחת בחירה

תפריט **Report** מאפשר לשנות את נוסחת הבחירה שתיושם בדוח. נוסחת בחירה מאפשרת לבצע סינון הרשומות שתוצגנה, מתוך Crystal Reports. פעולת נוסחת הבחירה דומה לזו של קטע WHERE במשפטי SQL, אך המבנה התחבירי שונה. תיבת הדו-שיח Record Selection Formula מאפשרת לסנן רשומות על ידי בחירת שדות וערכים רלוונטיים לפעולת הסינון. תיבת הדו-שיח בונה את משפט התוכנית המכיל את קריטריון הבחירה באופן אוטומטי. תרשים 29.15 ייראה כיצד לשנות את קריטריון הבחירה כך שייבחרו רק מחברים שנולדו במהלך שנת 1950 או אחריה.

זכור ששינוי קריטריון הבחירה הוא פעולה יעילה פחות משינוי מערך הרשומות שעליו מתבסס תהליך הבחירה. שינוי ביטוי הבחירה לא ישנה את מספר הרשומות שתיכללנה במערך הרשומות, אלא רק את מספר הרשומות שתוצגנה. בדוגמה, מערך הרשומות יכיל את כל רשומות הסופרים, אך תוצגנה רק רשומות הסופרים שנולדו מאז 1950.



תרשים 29.15: שימוש בחלון Formula Editor (עורך הנוסחאות) מאפשר להרכיב ביטויים הכוללים שדות, פונקציות ואופרטורים

שימוש בפקד Crystal Reports

לאחר שתשלים את עיצוב הדוח, יעמוד לרשותך קובץ RPT שיימצא בדיסק. תוכל להשתמש בקובץ זה בשילוב עם פקד Crystal Reports לצורך הצגת הדוח ביישום Visual Basic.

הערה:

אם תשתמש בדוח שנוצר בעזרת Crystal Reports ביישום. יש להפיץ את היישום כשהוא מלווה בכל קבצי העזר הדרושים לעבודה עם הדוח במערכת המשתמש.

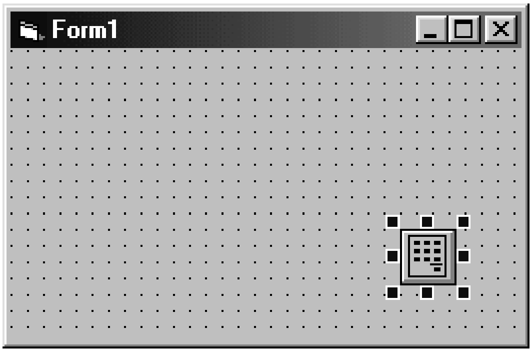
משום שפקד Crystal Reports אינו מתוצרת Microsoft, הוא אינו מוצע בארגז הכלים של Visual Basic כברירת מחדל. אם תרצה להוסיף אותו, בחר את הפקודה **Components** מתפריט **Project** או הקש Ctrl+T להצגת תיבת הדו-שיח, וסמן את Crystal Report Control 4.6.

כדי להדגים את פעולת הפקד, ניעזר בו להצגת הדוח שיצרנו בסעיף הקודם. תחילה יהיה עליך ליצור פרויקט חדש מסוג Standard EXE ולהוסיף את פקד Crystal Reports לארגז הכלים. שרטט פקד Crystal Reports על הטופס. שים לב שהפקד יוצג כשהוא בגודל סמל, כמתואר בתרשים 29.16.

הגדרת קובץ הדוח

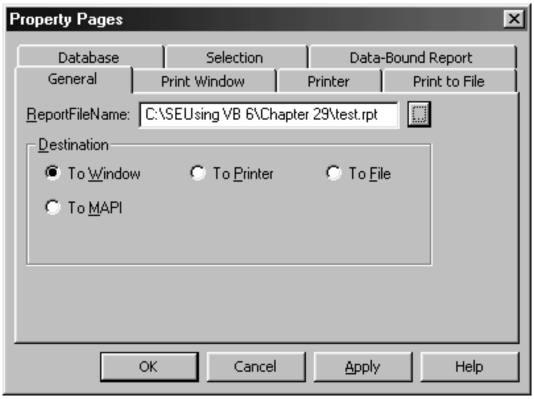
המאפיין החשוב ביותר שיש להגדיר בעת העבודה עם פקד Crystal Reports הוא ReportFileName, שיכיל את נתיב הגישה לקובץ RPT שיצרת בעזרת Crystal Report Designer. אם תרצה להשתמש בפקד להצגת דוחות שונים, כדאי אולי שתגדיר מאפיין זה בזמן ריצה, באופן הבא:

```
CrystalReport1.ReportFileName = App.Path & "\Test.Rpt"
```



תרשים 29.16: פקד Crystal Reports **אינו** מוצג על הטופס בזמן ריצה. הדוחות מוצגים בחלון נפרד

אך לצורך עבודה על דוח הדוגמה תוכל להגדיר את המאפיין בעת עיצוב הדוח. הגדר את המאפיין באמצעות לחיצה ימנית על הפקד ובחירה בפקודה Crystal Properties מהתפריט שיוצג, כדי להציג את גיליון המאפיינים של הפקד. כרטיסיית General (הנראית בתרשים 29.17), תאפשר להגדיר את מיקום קובץ RPT. בכרטיסיה זו תוכל גם לקבוע לאן יופנה פלט הדוח: מדפסת, תצוגה מקדימה, קובץ או הודעת דואר אלקטרוני.



תרשים 29.17: גיליון המאפיינים של פקד Crystal Reports **יאפשר** להגדיר קובץ שעליו יתבסס הדוח ויעד שאליו יופנה פלט הדוח

הגדרת המאפיין ReportFileName היא הפעולה המינימלית הנדרשת להגדרת פקד Crystal Reports. לאחר שתגדיר את המאפיין תוכל לכתוב את שורת הקוד היחידה הדרושה להפעלת הדוח ולנסות את הדוח על ידי הפעלה התוכנית.

הגדרת פרמטרים אופציונליים

המאפיין ReportFileName הוא אומנם היחיד שאותו יש להגדיר כדי שתוכל להציג את הדוח, אך ייתכן שתרצה להשתמש במאפיינים אופציונליים. SelectionFormula הוא הראשון. הוא מאפשר להגביל את מספר הרשומות שתיכללנה בדוח. אופן פעולתו דומה לזה של קטע WHERE במשפט SQL, אך הוא נעזר בתחביר שונה לצורך הגדרת הקריטריונים המיושמים בעת בחירת הרשומות (התחביר המיושם זהה לזה שתואר בסעיף **התאמה אישית של הדוח**). להלן דוגמה לשימוש בקוד להגדרת מאפיין:

```
CrystalReport1.SelectionFormula = "{Authors, Year Born} >= 1955"
```

ניתן גם לבנות קריטריונים מורכבים על ידי שימוש באופרטורים מסוג And ו-Or.



אזהרה:

אם תגדיר את **מאפיין** SelectionFormula בעת עיצוב הדוח, כל נוסחה שתציב **במאפיין** בעת הגדרת פקד Crystal Reports תתפרש **כקריטריון** נוסף שייכלל במסנן הבחירה.

CopiesToPrinter הוא מאפיין אופציונלי נוסף, אשר מקל על הדפסת מספר עותקים ב"מכה אחת". במאפיין זה ניתן להציב כל ערך מספרי שלם.

המאפיין האחרון שאליו נתייחס, הוא DataFiles. מאפיין זה אינו זמין לשימוש בעת עיצוב הדוח. המאפיין מכיל את שם קובץ מסד הנתונים שעמו יעבוד הדוח. ייתכן שתתהה מדוע יש להגדיר שוב את שם הקובץ, מאחר שהגדרת אותו כבר בעת העיצוב. אך כאשר עיצבת את הדוח, שם הקובץ אוחסן יחד עם הנתיב המפורט המשמש לגישה אליו, שהיה מבוסס על מבנה עץ התיקיות הקיים במחשב שלך, וסביר להניח שמבנה התיקיות במחשבי המשתמשים יהיה שונה.

מאפיין DataFiles הוא למעשה מערך, שמספרו הסידורי של הפריט הראשון בו הוא 0. אם הדוח מבוסס על יותר ממסד נתונים אחד, יש להציב ערך בכל אחד מפריטי מערך DataFiles. אך רוב הדוחות יתבססו על מסד נתונים אחד. שורת הקוד הבאה תדגים כיצד יש להגדיר את ערך מאפיין DataFiles. שורת הקוד מתבססת על ההנחה שקובץ מסד הנתונים מאוחסן בתיקיה בה מאוחסן היישום שלך.

```
rptMember.DataFiles(0) = App.Path & "\Members.mdb"
```

הצגת הדוח

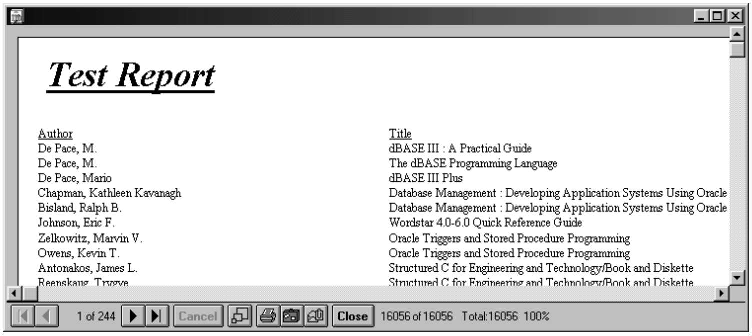
גם לאחר שתוסיף את פקד Crystal Reports לטופס היישום ותגדיר את המאפיינים, יש להודיע ל- Crystal Reports מתי יהיה עליו להציג או להדפיס את הדוח, על ידי הוספת שורת קוד ליישום. הדוח יודפס תוך התייחסות לערכים שהצבת במאפיין ReportFileName ובמאפיינים האחרים שהגדרת. אם כללת בדוח הגדרה של תצוגה מקדימה, התצוגה תוצג בחלון כמו זה המוצג בתרשים 29.18. שורת הקוד הבאה תפעיל דוח על ידי שימוש בשיטת PrintReport :

```
CrystalReport1.PrintReport
```

שים לב ששם השיטה הוא אומנם PrintReport, אך היא תפנה את הפלט ליעד שאותו הגדרת קודם לכן.

הערה:

ניתן להדפיס דוח גם על ידי הצבת ערך 1 במאפיין Action של פקד Crystal Reports.



תרשים 29.18: הדוח הרצוי הוצג על המסך

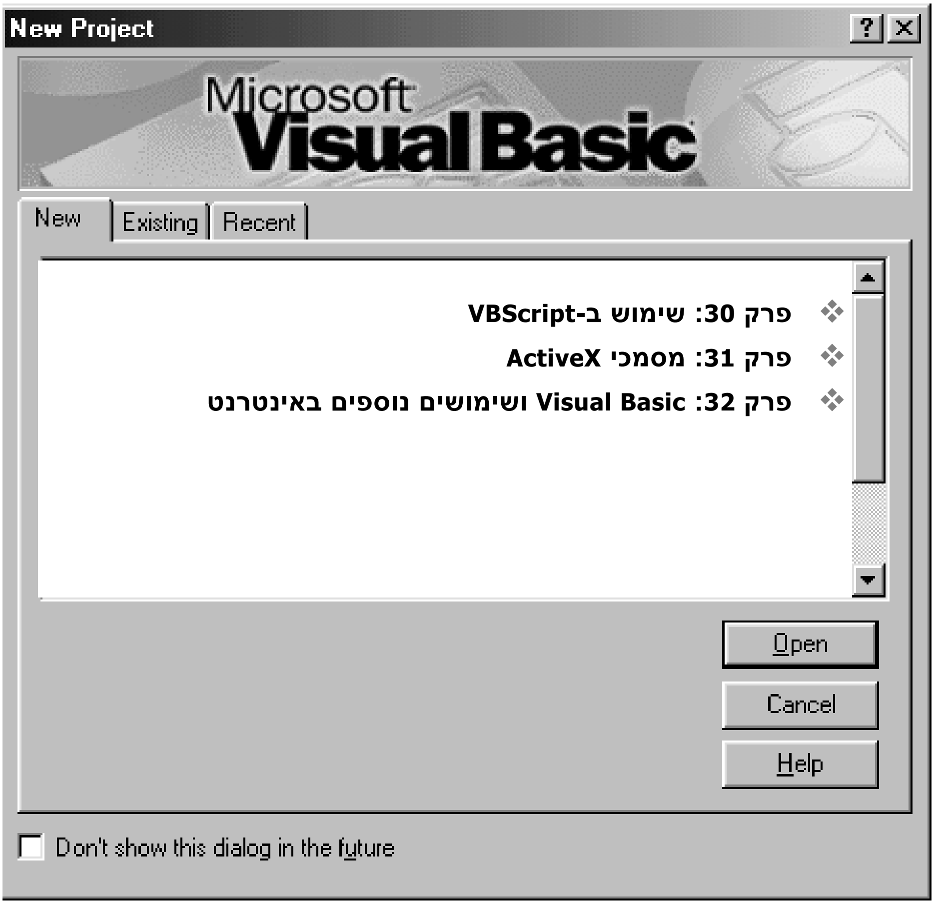
מכאן...

בפרק זה התנסית ב-Data Report שהוא אחד הכלים החדשים ב- Visual Basic 6.0. במהלך יצירת דוח פשוט, למדת על השיפורים שתוכל להכניס במראה הדוח, ועל שיטות למיכון פעולות הדפסה וייצוא דוחות. כמו כן סקרנו את הכלי להפקת דוחות הוותיק Crystal Reports. מידע נוסף תוכל למצוא בפרקים באים :

- מידע על שימוש ב-Excel וב-Word להפקת דוחות תוכל למצוא בפרק 22 **שימוש ב-OLE לשליטה על יישומים אחרים**.
- מידע כללי אודות מסדי נתונים תוכל למצוא בפרק 24 **יסודות מסד נתונים**.
- מבוא לשימוש בטכנולוגיית ADO לאחזור נתונים, תוכל למצוא בפרק 28 **גישה לאובייקטי נתונים באמצעות פקדי ActiveX**.

חלק 7

Visual Basic והאינטרנט



30

שימוש ב-VBScript

מה בפרק זה?

- מבוא לשפת VBScript
- כלים לעבודה עם VBScript
- שפת VBScript
- VBScript ב- Internet Explorer
- Windows Scripting Host

פרק זה פותח את החלק הדן ב-Visual Basic ב-Web. ‏Web הוא אחד הנושאים ה"חמים" בעולם המחשבים של ימינו. רשת ה-Web - קיימת אומנם מזה שנים רבות, אך עד לפני שנים מועטות היא נסתרה מעיני הציבור הרחב, ושימשה בעיקר את המגזר האקדמי. כיום נוצר רושם שלכל אחד יש דואר אלקטרוני (E-Mail) ואתר Web. הפרק ילמד אותך דרכים ליישום הידע ב-Visual Basic במסגרת ה-Web. הוא גם מהווה מבוא לשפת VBScript, שהיא נגזרת Visual Basic המשמשת לתכנות בסביבת Web.

מבוא לשפת VBScript

פרט לגרסת כלי הפיתוח העצמאית, Visual Basic מוצעת במספר גרסאות נוספות. אחת הגרסאות היא (VBA) Visual Basic for Applications. גירסה נוספת, שחשוב שלא להתבלבל בינה לבין VBA היא Visual Basic Scripting Language המוכרת בשמה המקוצר VBScript. טבלה 30.1 תביא תיאור מקוצר כל אחת מהגרסאות.

טבלה 30.1: השוואה בין Visual Basic, ‏VBA ו-VBScript

שפה	תיאור
Visual Basic	סביבת פיתוח עצמאית אשר מאפשרת למשתמש בה להדר קבצי ביצוע, פקדי ActiveX וקבצי DLL
VBA	גרסת Visual Basic המיועדת לשימוש ביישומים כגון Excel, ‏Access ו-Visio. החל בגרסת Office 97, השפות Visual Basic ו-VBA נעזרות באותה סביבת פיתוח משולבת - IDE (Integrated Development Environment)
VBScript	גירסה מצומצמת מאוד של Visual Basic שבשימוש יישומים כגון Internet Explorer, ‏Internet Information Server, ‏Microsoft Outlook ו-Windows Scripting server

עוצמת VBScript אינה מתקרבת כלל לעוצמות Visual Basic ו-VBA והיא גם אינה כוללת סביבת פיתוח משלה. מגבלות אלו הוטלו על השפה בעת עיצובה. אך עוצמתה אינה טמונה בשפה עצמה, אלא באופנים בהם ניתן להשתמש בה, למשל:

- עמודי Web - ניתן לשלב קוד VBScript המיועד לשימוש בצד הלקוח (Client-side VBScript Code) בעמודי Web, ועל ידי כך להקנות לעמודים אלה יכולות הדומות לאלו שביישומים.
- Active Server Pages (דפי שרת פעילים) - קוד VBScript המיועד לשימוש בצד השרת (Server-side VBScript Code) יכול לשמש ב- Active server Pages כדי לאפשר יצירת עמודים כאלה ושינוי תוכנם לפני החזרתו ליישום הלקוח.
- Windows Scripting Host - ניתן להריץ Script הכתוב ב- Visual Basic משורת הפקודה של DOS, כפי שמפעילים קבצי אצווה. יכולת זו מאפשרת למכן ביצוע מטלות מסוימות.

שיפורים ב-Web בעזרת VBScript

VBScript נועדה בתחילה לשימוש באינטרנט. האינטרנט היא רשת פיסית של מחשבים. אך בעת שאנשים חושבים על המושג אינטרנט, הם נוהגים להתייחס ליישומים מסוימים הפועלים בה ולא על הרשת עצמה. להלן כמה מיישומים אלה:

- **World Wide Web** - אוסף מסמכים אינטראקטיביים הניתנים להצגה באמצעות דפדפן (Web Browser). מסמכים כאלה יכולים להכיל מידע חדשותי, נתונים ותוכניות שאותן תוכל להוריד (Download) למחשב שלך.
- **E-Mail** - דואר אלקטרוני המשמש להעברת מסרים למשתמשים ספציפיים.
- **Newsgroups** - קבוצות דיון המשתמשות במבנה לוח מודעות שבהן ניתן לפרסם שאלות ותשובות העוסקות במיגוון נושאים רחב.
- **Chat** - תוכנות המאפשרות לקיים תקשורת חיה עם משתמשים אחרים המחוברים לרשת, תוך שימוש במסרים כתובים, בקבצי קול ובאותות וידאו.
- **העברת קבצים וגישה למחשבים מרוחקים** - העברת קבצים למחשבים מרוחקים (על ידי שימוש ב-FTP - File Transfer Protocol - פרוטוקול העברת קבצים) וגישה למחשב מרוחקים (על ידי שימוש בפרוטוקול Telnet).

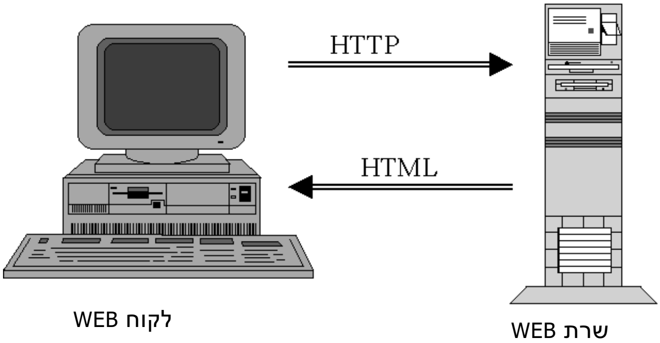
השימוש ב-World Wide Web הוא בוודאי אופן השימוש הפופולרי ביותר ב-Web. רעיון ה-World Wide Web נהגה על ידי Tim Berners-Lee כאמצעי לשיפור הזיכרון על ידי הכללת **קישורים** (Links) במסמכים. יישום הרעיון הוביל לפיתוח Global Hypertext Space שהוביל לפיתוח ה-Web המוכר כיום, המכיל מיליוני מסמכים. כל קישור מכיל הפניה למסמך מסוים, וגם מאפשר לעבור ישירות למסמך שאליו הוא מפנה (ומכאן מקור הכינוי Hyperlink - היפר קישור). המסמכים המקושרים ביניהם באמצעות היפר-קישורים מרכיבים "רשת" מידע הנגיש לכל מי שמחובר.

VBScript בשרת Web

סביר להניח שאתה יודע, מתוך התנסות אישית, כי ה-Web הקיים כיום עבר כברת דרך ארוכה מאז הימים שבהם הוא היה מורכב ממסמכי טקסט המקושרים ביניהם. כיום, אתרי Web מכילים קבצי מולטימדיה מגוונים ונתונים הקשורים למסדי נתונים, ואתרים כאלה יכולים לפעול כמעט כמו תוכניות רגילות. רבים טוענים שהשיפורים החשובים ביותר ב-Web הוכנסו בשרתים המשמשים בו.

שרת Web (Web Server) הוא מחשב שמעביר דפי Web שנדרשו על ידי דפדפנים. הבנת המושג **דרישה** (Request) היא המפתח להבנת הפעולה של שרתי Web. אנשים רבים חושבים שדפי Web מאוחסנים בשרת קבצים, אך זה אינו אופן הפעולה המיושם ב-Web. תפקיד שרת Web מתואר בתרשים 30.1.

נמחיש את ההבדלים על ידי דוגמת חיפוש מידע על ויליאם שייקספיר בספריה בה אתה מנוי. תוכל לגשת למדף עליו מונחים כתבי שייקספיר ולקחת ספר, ולחילופין תוכל להיעזר באחד מעובדי הספריה.



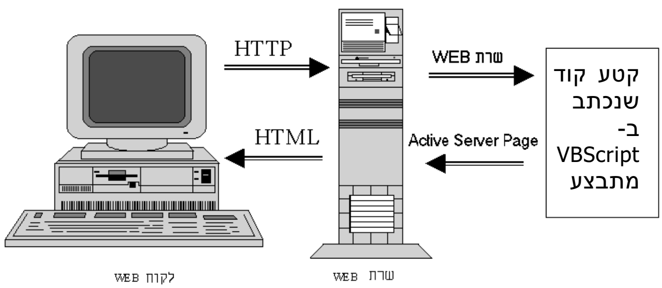
תרשים 30.1: כמענה לדרישה לקבל דף Web, שרת Web שולח נתוני HTML לדפדפן

כשאתה לוקח את הספר בעצמך, הספריה משמשת כשרת קבצים ואתה מבצע פעולת גישה לקובץ (ספר). כשתפנה לאחד מעובדי הספריה, הוא ישאל אותך שאלות הנוגעות מתוך הידע המקצועי שלו, ויציע הצעות המבוססות על התשובות שתענה. במצב כזה העובד מתפקד כשרת Web, בכך שהוא מבקר את הגישה שלך לקבצים (ספרים).

הערה:

ה"שפה" שבה "מדברים" ביניהם דפדפנים ושרתי Web נקראת Hyper Text Transfer Protocol (HTTP - פרוטוקול העברת היפר טקסט).

משום שכל הדרישות מועברות לשרת Web, מתאפשר לו לשנות את המידע שמוחזר לדפדפן באופן הרצוי. אחת הדוגמאות היא הכללת נתונים הלקוחים ממסד נתונים בדפי Web. ניתן לבצע פעולה כזו על ידי שימוש בתוכנית שתפעל בשרת, תטפל בנושא הגישה למסד הנתונים, ותחזיר נתונים תוך שימוש במידע שיהיה מובן לדפדפן. האופן בו Microsoft מספקת יכולות כאלו למפתחי Visual Basic מוכר בשם Active Server Pages (ASP - דפי שרת פעילים). טכנולוגיית ASP מוצגת בתרשים 30.2.



תרשים 30.2: השרת מבצע Script שב-ASP, לפני השבת התוצאות למשתמש

טכנולוגיית ASP היא טכנולוגיה חדישה וחזקה, שחשיבותה רבה מספיק כדי שנקדיש לה פרק שלם בספר זה, נספח 4 **דפי שרת פעילים**.

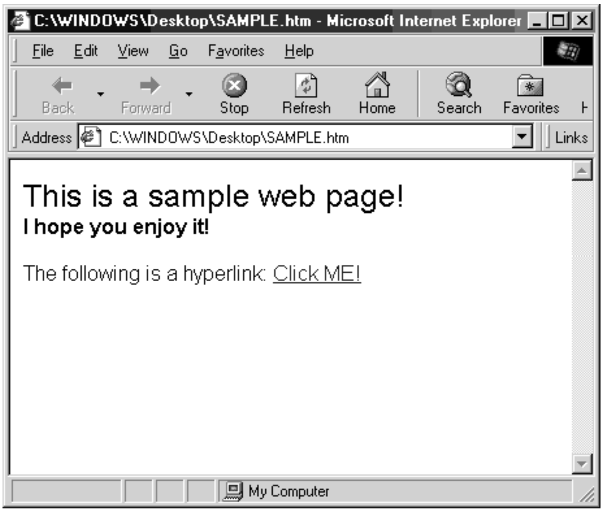
VBScript בדפדפן

בנוסף לשימוש בשפת VBScript בשרת Web, ניתן להשתמש בקוד VBScript גם במחשב המשתמש. כדי שיתאפשר לך להבין כיצד מתבצעת פעולה כזו, תידרש להבין מעט יותר בשפה שבה כתובים דפי Web : שפת HTML.

שפת **HTML** ששמה מורכב מראשי התיבות Hypertext Markup Language מורכבת משילובי תווים המהווים קודי עיצוב, המאפשרים למחברי מסמכים להגדיר Hyperlinks ולעצב טקסט. שפת HTML היא פשוטה יחסית להבנה, גם עבור משתמשים חדשים, כפי שניתן לראות בדוגמה הבאה :

```
<FONT SIZE = "+2"> This is a sample web page! </FONT><BR>
<B> I hope you enjoy it! </B> <BR><BR>
The following is a hyperlink: <A HREF = "http://www.hod-ami.co.il"> Click me! </A>
```

HTML מכילה את הטקסט שיעוצב ואת ההגדרות שתשמשנה לעיצוב הטקסט. תוכנת הצגה מיוחדת הקרויה **דפדפן** (Browser) מפרשת את קודי העיצוב ומציגה את עמוד Web המוגדר על-ידם. בתרשים 30.3 תראה עמוד Web שנוצר על בסיס קובץ HTML שהובא בדוגמה, כשהוא מוצג בדפדפן.



תרשים 30.3: לכל מי שהשתמש במחשב בשנה-שנתיים האחרונות **יצא** להיתקל בדפדפן

מפתחי Visual Basic טובים לא יתקשו להבין את יסודות שפת HTML. הדוגמה שהובאה לעיל הכילה כמה קודי עיצוב המכונים **תגים** (Tags) בטרמינולוגיית HTML. לרוב האלמנטים העיצוביים בשפת HTML הוגדרו נקודות התחלה וסיום שמצוינות על ידי תגים מתאימים. לדוגמה התגים <B> ו-</B> מורים לדפדפן שאת הטקסט ביניהם יש להציג בכתב מודגש.



טיפ:

תוכל ללמוד אודות שפת HTML גם על ידי הצגת קוד המקור של דפי Web החביבים עליך. להצגת קוד המקור עמוד בעת שימוש ב-Internet Explorer, לחץ לחיצה ימנית על שטח ריק בעמוד ובחר מהתפריט תלוי ההקשר שיוצג, את הפקודה **View Source**.

כיום מוצעים ספרים רבים ומקורות מידע מקוונים רבים, שמהם תוכל ללמוד אודות שפת HTML. טבלה 30.2 תפרט את האלמנטים השכיחים ביותר בשפת HTML.

טבלה 30.2: תגים שכיחים בשפת HTML

תיאור	תג
פס הפרדה אופקי	<HR>
מעבר שורה	
הדגשה	<B> </B>
טקסט נטוי	<I> </I>
קו תחתי	<U> </U>
מרכוז טקסט	<CENTER> </CENTER>
הוספת Hyperlink (או Anchor - עוגן)	<A HREF=url> link text </A>
הוספת קובץ מסוג JPEG או GIF	<IMG SRC=imagefile ALT=tooltip text>
לציון תחילה וסיום מסמך	<HTML> </HTML>
לציון תחילה וסיום ראש המסמך (Header)	<HEAD> </HEAD>
כותרת הדף המוצגת בראש המסמך	<TITLE> </TITLE>
לציון נקודות התחלה וסיום גוף המסמך (Body)	<BODY> </BODY>
לציון תחילה וסיום טבלה (Table)	<TABLE> </TABLE>
ביחד עם תג <TABLE> מגדיר שורה (Row) בטבלה	<TR> </TR>
ביחד עם תג <TR> מגדיר עמודה (Column) בטבלה	<TD> </TD>
מגדירים נקודות התחלה וסיום טופס (Form)	<FORM action=url method=method name> </FORM>
פריט קלט בטופס	<INPUT type=type value=value name=name>
מגדירות מאפיינים שונים של הגופן הפעיל, כגון צורת אות וגודל אות	<FONT> </FONT>

הערה:

Internet Explorer 4.0 **תומך גם בשפת HTML דינמית** (Dynamic HTML - DHTML), שמציעה למפתחים מיגוון יכולות רחב יותר בפיתוח דפי HTML, היות והיא מאפשרת גישה לתגי HTML מתוך קוד תוכנית.

קודי HTML שפורטו בטבלה 30.2 הם רק חלק מכלל הקודים שמוגדרים בשפה זו. בנוסף ניתן לקנן קודים באופנים שונים, כדי להפיק תוצאות שונות. שפת HTML אומנם כוללת מיגוון יכולות מעניינות, אך היא חסרה את העוצמה של שפות תכנות רגילות. למעט יכולת לאפשר מעבר למסמכים אחרים, ולהעברת נתונים מהטופס לשרת, שפת HTML הסטנדרטית אינה מאפשרת לבצע פעולות כלשהן. אך ניתן לצרף לקבצי HTML קוד Script, שירחיב מעט את מיגוון היכולות. ניתן לשלב Script בשפות VBScript ו-JScript ב-HTML, כאשר קוד כזה יהיה תחום בתגים <SCRIPT>. קטע הקוד הבא מדגים שימוש בשפת VBScript להצגת תיבת הודעה:

```
<SCRIPT Language = "VBScript">
<! - -
Dim nAnswer
nAnswer = MsgBox ("Would you like to visit my site?", vbYesNo, "Hello!")
If nAnswer = vbNo Then
   MsgBox "Well, then I Will send you somewhere else!"
   Window.Location = "http://www.nowhere.com"
end if
<--
</SCRIPT>
<FONT SIZE = "+2"> This is a sample web page! </FONT><BR>
<B> I hope you enjoy it! </B> <BR><BR>
The following is a hyperlink: <A HREF = "http://www.hod-ami.co.il"> Click me! </A>
```

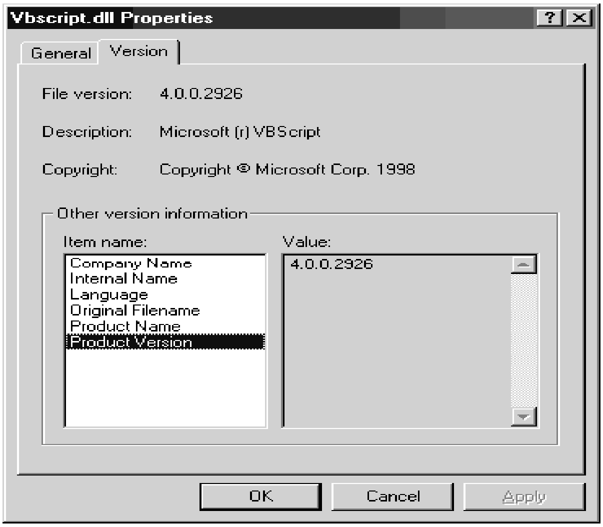
מסמך HTML שהוצג כאן מכיל Script הכתוב ב-VBScript בתחילתו. כאשר הדפדפן מתחיל לפרש את תגי HTML שבבקובץ, הוא נתקל גם ב-Script ומבצע אותו. בהמשך הפרק יובאו פרטים נוספים על שימוש בשפת VBScript בדפי Web.

כלים לעבודה עם VBScript

כדי לעבוד עם VBScript נדרשים מספר כלים. הסעיפים הבאים יכללו תיאורים קצרים על מספר פריטים שיקלו עליך את העבודה עם VBScript.

VB Scripting Engine

קוד Script מתורגם על ידי Scripting Engine (מנגנון טיפול ב-Script) כלשהו ומבוצע על-ידו. Scripting Engine המשמש לעבודה עם VBScript הוא קובץ ששמו VBSCRIPT.DLL שמאוחסן בתיקיה Windows\System\. VBScript מותקנת במסגרת התקנת יישומים הנעזרים בה, כגון Internet Explorer, Outlook, Internet Information Server, או Windows Scripting Host. במשך השנים שוחררו מספר גרסאות VBScript. הדוגמאות שתוצגנה במהלך פרק זה לא תעבודנה עם גרסאות שלפני גירסה 3.0. אם תרצה לבדוק איזו גרסת VBScript מותקנת אצלך, לחץ לחיצה ימנית על קובץ VBScript.dll בסייר Windows, ובחר את הפקודה **Properties** מהתפריט תלוי ההקשר שיוצג לפניך, כמתואר בתרשים 30.4.



תרשים 30.4: ודא שאתה משתמש בגרסת VBScript העדכנית ביותר, על ידי עיון במאפייני קובץ VBSCRIPT.DLL

אינדיקציה נוספת לכך שגרסת VBScript המותקנת אינה עדכנית, מתקבלת בצורת הודעת שגיאה המוצגת בעת ניסיון לשימוש בקבוע פנימי (Intrinsic Constant), מפני שגרסת VBScript הראשונה לא תמכה בקבועים כאלה. אם תרצה להתקין את הגירסה העדכנית ביותר, תוכל להורידה מהאתר http://www.microsoft.com/vbscript. מאתר זה ניתן להוריד תיעוד של שפת VBScript, ואת הגרסאות העדכניות ביותר של **מנגנוני הטיפול ב-Scripts** (Scripting Engines) שאותם תוכל לשלב באתר.

יישום מארח

יישומים שבהם ניתן להשתמש ב-VBScript מכונים יישומים **מארחים** (Hosts). השכיחים ביותר הם Internet Explorer ו- Internet Information Server. גם Outlook משתמשת ב-VBScript כשפה לכתיבת מאקרוס. Microsoft החלה לאחרונה בשיווק מוצר המכונה Windows Scripting Server המאפשר להשתמש ב-VBScript על פלטפורמות רגילות מבוססות Windows, כדוגמת Windows 95 ו- Windows NT.

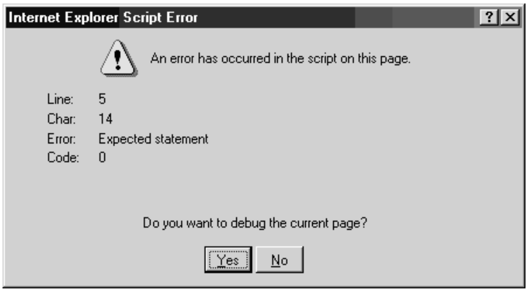
הערה:

Internet Explorer משמש כמארח עבור קוד VBScript שפועל בשולחן העבודה של המשתמש. כיום לא ניתן להשתמש ב-VBScript בעת עבודה עם Netscape Navigator. אך ניתן להשתמש בקוד VBScript ב- Active Server Pages שפועלים על Internet Information Server ומאפשרים ליצור דפי HTML שאינם תלויים בסוג השרת או דפי HTML המיועדים לשימוש בשרת מסוג מסויים, תוך כדי פעולת היישום.

שפת VBScript אומנם פועלת באופן זהה על פלטפורמות שונות, אך יש לזכור שיישומים מארחים שונים חושפים אובייקטים שונים בפני VBScript, כך שיש פקודות VBScript שתפעלנה בעת עבודה עם Internet Explorer ולא תפעלנה בעת עבודה עם Internet Information Server או להיפך.

עורך טקסט

אנשים רבים הנוהגים להשתמש בשפת VBScript בדפי Web נוהגים להשתמש בעורכי טקסט פשוטים לשם עריכת ה-Script, זאת היות של שפת VBScript (עדיין) אין סביבת פיתוח משולבת משלה. מכך משתמע שכל הכלים המועילים שבהם אתה נוהג להשתמש בעת עבודה ב-Visual Basic אינם זמינים לשימוש בעת עבודה עם VBScript. עליך להקפיד על שימוש בכתיב נכון בטקסט שאתה מקליד ועל עיצוב נכון שלו. כיום מוצעות תוכנות מתקדמות כגון HomeSite אשר מסייעות בהזנת הטקסט, בכך שהן מיישמות שיטות הצגת טקסט תוך שימוש בצבעים שונים, והן כוללות לחצנים המאפשרים להזין פקודות שכיחות בעזרת לחיצה יחידה על לחצן.



תרשים 30.5: כאשר מאותרת שגיאה בקוד, Internet Explorer מציג את מספר השורה שבה אותרה השגיאה

בהקלדת טקסט, ניתן להשתמש בתוכנות פשוטות לעריכת טקסט, כגון Notepad, אך אני ממליץ להשתמש בתוכנת עריכה שמציגה מספרי שורות כגון Visual Inter Dev. זאת היות וכאשר מתרחשת שגיאה, Internet Explorer מחזיר את מספר השורה שבה אירעה השגיאה, כמתואר בתרשים 30.5.

אם תעבוד כאשר הדפדפן ועורך הטקסט פתוחים במקביל, תוכל לעבור ביניהם בקלות ובמהירות. זהו אחד האופנים בהם אתה יכול לנפות שגיאות: ערוך את הקוד ושמור אותו, ולאחר מכן לחץ בדפדפן על לחצן Refresh.

כלי Web מתקדמים

אם יימאס לך לעבוד אך ורק עם עורך טקסט, תוכל לנסות לעבוד עם כלים מתקדמים יותר, כדוגמת אלה המפורטים להלן:

- **ActiveX Control Pad** - תוכנה מתוצרת Microsoft שמופצת בחינם. היא אינה מתקדמת מאוד, אך מאפשרת להציב ערכים במזהה Class ID של פקדי ActiveX.
- **Microsoft FrontPage** - תוכנה ליצירת דפי Web. FrontPage מכילה Script Wizard שיוצר קוד VBScript, ומקל על העבודה עם פקדי ActiveX ושיטותיהם.
- **Microsoft InterDev** - מוצר נוסף של Microsoft, המיועד יותר לשימוש מפתחים של דפי Web הקשורים למסדי נתונים. התוכנה יוצרת קוד VBScript.
- **Internet Client SDK** - **ערכת פיתוח תוכנה** (Software Development Kit) הניתנת להורדה מאתרי Microsoft. הערכה כוללת דוגמאות קוד, חומר הדרכה ותוכנות עזר לעבודה בסביבת Web.

שפת VBScript

כאשר משווים בין VBScript לבין Visual Basic הסטנדרטית מתברר כי VBScript היא גירסה שלדית כמעט "עירומה" של Visual Basic. אך חשוב לזכור שהיא נועדה להיות כזאת, עוצמת Visual Basic נובעת מהסביבה בה ניתן להשתמש בה, ומהאובייקטים שבהם היא תומכת ולא מהשפה עצמה. כך שלמי שרגיל לעבוד עם Visual Basic יידרש שינוי מחשבתי מסוים כדי לעבוד עם VBScript.

עבודה עם משתני Variant בלבד

ההבדל הבולט ביותר, כנראה, שקיים בין Visual Basic לבין VBScript הוא העובדה שכל המשתנים המשמשים בשפת VBScript הם מסוג Variant. להלן כמה דוגמאות של שורות חוקיות להצהרה על משתנים בשפת VBScript:

```
Dim sLastName
Dim nCounter
Dim myArray(2,5)
```

VBScript אינה תומכת במילת המפתח As שמשמשת ב- Visual Basic להצהרה על סוגי משתנים. ניסיון לשימוש במילה As ב-Script יגרום לשגיאה. חוק "Variant בלבד" תקף גם לגבי משתנים המשמשים בתת-שגרות ופונקציות:

```
Function CalcInterest(P,R,T)
      CalcInterest = P*R*T
End Function
```

הפרמטרים של הפונקציה CalcInterest והערך המוחזר על-ידה הם מסוג Variant, כך שקטע הקוד שקרא לפונקציה יכול היה להציב את המחרוזת "Hello" באחד הפרמטרים מבלי לגרום לשגיאה תחבירית, אך הצבת ערך כזה היתה גורמת לשגיאת Type Mismatch (אי התאמה בין סוג משתנה לבין פעולה) בעת פעולת הכפל. מסיבה זו אני ממליץ בחום להקפיד ליישם את כללי מתן השמות למשתנים, הנהוגים ב- Visual Basic. ניתן גם לשלב הצהרת Option Explicit בקטע Header של דף Web, כדי לאלץ להצהיר על משתנים.

VBScript גם תומכת בכמה פונקציות המשמשות לזיהוי סוג הערך המאוחסן במשתנה. פונקציות אלו תפורטנה בטבלה 30.3.

טבלה 30.3: זיהוי סוגי משתנים בשפת VBScript

פונקציה	ייעוד
VarType(varname)	מחזירה קבוע המציין סוג
TypeName(varname)	מחזירה תיאור סוג
Is()...function	מבצעת בדיקות לגילוי סוגי ערכים ותנאים מסוימים

קטע הקוד הבא יציג את מחרוזת תיאור הסוג התואמת לכל פריט במערך מסוג Variant:

```
Dim varArray(5)
Dim i
varArray(1) = "This is a string"
varArray(2) = 123.456
varArray(3) = "#12/25/1998#"
varArray(4) = 10
varArray(5) = Null

Dim i = 1 to 5
  MsgBox "Array element " & i & " is of type " & TypeName(varArray(i))
Next
```

שים לב לאופן השימוש בפקודה For בהצגת המערך. בשפת VBScript לא ניתן לציין שם משתנה לאחר פקודת Next. פעולה כזו תגרום לשגיאה.

גישה למערכת הקבצים

VBScript כוללת מספר פונקציות שמחליפות פונקציות רגילות של Visual Basic או מבצעות את הפעולות המבוצעות על ידי אותן פונקציות אך באופן שונה, המיועדות להבטיח ש-Scripts לא יזיקו למחשב שלך. תחום הגישה לקבצים הוא אחד התחומים שבו קיימים הבדלים ברורים ביניהן. ל-VBScript אין פקודות לביצוע פעולות קלט ופלט של קבצים, כדוגמת אלו שתוארו בפרק 21. בשפת VBScript פעולות הגישה לקבצים מבוצעות על ידי שימוש בשיטות אובייקטים מיוחדים. האובייקטים המשמשים לעבודה עם קבצים בשפת VBScript מפורטים בטבלה 30.4.

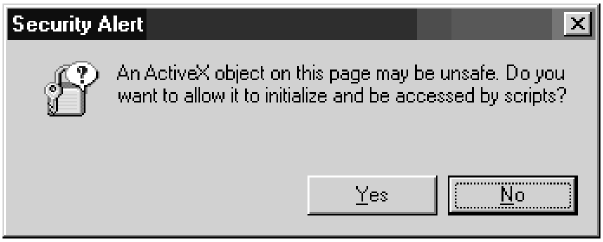
טבלה 30.4: אובייקטי VBScript המשמשים לעבודה עם קבצים

אובייקט	תיאור
File	קובץ במחשב מקומי
Folder	תיקיה (תיקיה) במחשב מקומי
Drive	כונן במחשב מקומי
TextStream	אובייקט המספק יכולות קריאה/כתיבה סדרתיות
FileSystemObject	אובייקט המייצג מערכת קבצים שלמה, משמש להפנייה אל כל האובייקטים שפורטו בטבלה

לביצוע פעולות גישה לדיסקים וקבצים יהיה עליך להשתמש בשיטות ובמאפייני האובייקטים שפורטו בטבלה 30.4. האובייקט החשוב ביותר מבין אלה הוא FileSystemObject. יהיה עליך להשתמש בשיטה CreateObject של VBScript כדי ליצור ייחוס לאובייקט כזה, כפי שהדבר נעשה בדוגמה הבאה:

```
Set objFs = CreateObject("Scripting.FileSystemObject")
```

אובייקט מסוג זה מציע למפתחים מרושעים דרכים שתאפשרנה להם לפגוע במערכת שלך. זו הסיבה שבעטייה בכל פעם ש-VBScript מבצעת ניסיון גישה לאובייקט חיצוני, Internet Explorer מציג הודעת אזהרה, כמו זו המוצגת בתרשים 30.6.



תרשים 30.6: Internet Explorer תומך בכמה הגדרות אבטחה (Security Settings) המיועדות למנוע מ-Scripts ופקדים מזיקים לפעול על המחשב שלך

תוכנית 30.1 מציגה תאריך ושעה שבהם משתמש ביקר בפעם האחרונה בדף Web. היא מנהלת מעקב אחר נתונים על ידי קובץ LASTVISIT.TXT שבתיקיה Windows. פתח את הקובץ ב- Internet Explorer ולחץ מספר פעמים על Refresh כדי לנסותו.

תוכנית 30.1: VBSFILES.HTM - גישה למערכת הקבצים על ידי Internet Explorer.

```
<HTML>
<HEAD>
<SCRIPT Language = "VBScript">
<!--OPTION EXPLICIT
	Dim objfs	'FileSystemObject Object
	Dim objfile	'File Object
	Dim objTs	'TextStream Object
	Dim sInfoFile	'Path to the info file
	Dim sInfo	'Variable used to store information
	'Create a reference to the local file system
	Set objFs = CreateObject("Scripting.FileSystemObject")
	'Determine path to file in the windows directory
	sInfoFile = objFS.GetSpecialFolder(0) & "\LASTVISIT.TXT"
	'If file exists then read a line of text
	If objfs.FileExists(sInfoFile) Then
		Set objfile = objfs.GetFile(sInfoFile)
		Set objts = objfile.OpenAsTextStream(1) 'Reading = 1, Writing = 2
		sInfo = objts.ReadLine()
		objts.close
	Else
		sInfo = "You haven't visited this web site before!"
	objfs.CreateTextFile sInfoFile,True
	Set objfile = objfs.GetFile(sInfoFile)
	End If
	'Write message to the browser window
	Document.Write sInfo & "<BR>"
	'Store new message in the file for next time
	sInfo = "Your last visit was <B>" & Now & "</B>. Welcome back!"
	Set objts = objfile.OpenAsTextStream(2) 'Reading = 1, Writing = 2
	objts.WriteLine(sInfo)
	objts.close
	Set objfs = Nothing
-->
</SCRIPT>
</HEAD>
</HTML>
```

VBScript ב- Internet Explorer

הסעיפים הבאים יילמדו כיצד לשלב קוד VBScript המיועד לפעול ביישום הלקוח (Client-Side VBScript) בדפי Web. VBScript יכולה לפעול גם בשרת, כפי שיתואר בפרק 30. קוד הכתוב בשפת VBScript שפועל בשרת אינו תלוי בסוג דפדפן מסוים, אך קוד VBScript המיועד לפעול ביישום הלקוח דורש שאצל המשתמש יהיה מותקן דפדפן מסוג Internet Explorer. לעת עתה, דפדפני חברת Netscape אינם תומכים בשפת VBScript, כדאי שתזכור עובדה זו בעת כתיבת קוד בשפת VBScript.

אירועים ושגרות

בעת כתיבת קוד Visual Basic נהוג להגיב על התרחשות אירועים על ידי מיקום קוד בשגרות אירוע (Event Procedures). ניתן לכתוב שגרות אירוע גם בשפת VBScript, אך כתיבתן אינה פשוטה כפי שהיא בעת שימוש ב- Visual Basic, מפני שהשפה אינה תומכת במשפטי ההצהרה הדרושים לשם כתיבתן.

כדוגמה נתייחס לקטע HTML, שמגדיר שני פריטי קלט בטופס, תיבת קלט (תיבת טקסט) ולחצן:

```
<INPUT Type="text" Name="txtLastName" Value="Smith" Size=20>
<INPUT Type="Button" Name="cmdCalculate" Value="Preform Calculation">
```

משום שהגדרת שמות לשני פריטי קלט אלה, כל שיהיה עליך לעשות כדי לכתוב שגרת אירוע לטיפול בהם, הוא לכתוב שיגרה (Subroutine) בשפת VBScript תוך שימוש בשמות האלמנטים ובפרמטרים המתאימים. לדוגמה, על ידי כתיבת שיגרה בשם cmdCalculate_OnClick( ), תוכל לגרום ל- Internet Explorer לבצע קטע קוד מסוים כתגובה לאירוע לחיצה על הלחצן. תוכנית 30.2 משתמשת באירועי VBScript לחישובי ריבית פשוטים.

תוכנית 30.2: SIMPLEVBS.HTM - שימוש ב-VBScript לגישה לפריטים בדפי Web.

```
<HTML>
<HEAD><TITLE>Simple example of VBScript</TITLE></HEAD>

<SCRIPT Language = "VBScript">
<!--

Function CalcInterest(P,R,T)
        CalcInterest = P * R * T
End Function

Sub cmdCalculate_OnClick()
        Dim lPrincipal
        Dim dblRate
        Dim nYears
        Dim cInterest
```

```
        lPrincipal = CLng(txtPrincipal.value)
        dblRate = CDbl(txtRate.value)
        nYears = CInt(txtTime.value)

        cInterest = CalcInterest(lPrincipal, dblRate, nYears)

        MsgBox "Your Interest is " & FormatCurrency(cInterest)
End Sub
Sub txtRate_OnChange()
        If CDbl(txtRate.Value) > 1 Then txtRate.Value = txtRate.Value / 100
End Sub

-->
</SCRIPT>
<BODY>
<BR>

Enter Principal: <INPUT Type="Text" Name="txtPrincipal" Value="100000"><BR>
Enter Int. Rate: <INPUT Type="Text" Name="txtRate" Value="0.08"><BR>
Time in Years: <INPUT Type="text" Name="txtTime" Value="20"><BR>

<INPUT Type="Button" Name="cmdCalculate" Value="Calculate Interest">

</BODY>
</HTML>
```

הערה:

הקוד שבתוכנית 30.2 כולל התייחסות לפריטי קלט בטפסי HTML, שעבורם מוגדרים מאפיינים ושיטות השונים מאלה המוגדרים עבור אובייקטים VBScript. לדוגמה, הקוד פונה למאפיין Value של תיבת הקלט במקום למאפיין Text. בהמשך הפרק נביא דוגמת קוד נוספת שבה תראה שניתן לשלב בדפי Web פקדים שהשיטות והמאפיינים שלהם יהיו יותר מוכרים לך.

בדף Web שבתוכנית 30.2, דפדפן Internet Explorer יכול לזהות את שגרת האירוע שאותה הוא אמור להפעיל, לפי השם שהוגדר עבורה. אך באותה קלות תוכל גם לפנות לשגרת אירוע Click אחרת בכך שתשלב את שמה בקטע HTML שמתייחס ללחצן:

```
<INPUT Type="Button" Name="myButton" Value="Calculate" OnClick="MyProcedure">
```

זו שיטה יעילה לטיפול בתמונות בדפי Web. נתייחס לרגע לשיטה הרגילה להגדרת קישורים לתמונות:

```
<A HREF="http://www.newsite.com/"> <IMG SRC="myimage.gif"></A>
```

המבנה התחבירי הזה יבצע את הפעולה הרצויה, אך הוא יאפשר רק לעבור לדף אחר. ציון שם שגרת אירוע Click בתג IMG, יאפשר גמישות רבה יותר. שגרת האירוע תוכל להסתייע בקוד VBScript כדי לבצע מעבר לאתר אחר, או לצורך ביצוע כל פעולה אחרת שאותה תהיה מעוניין לבצע:

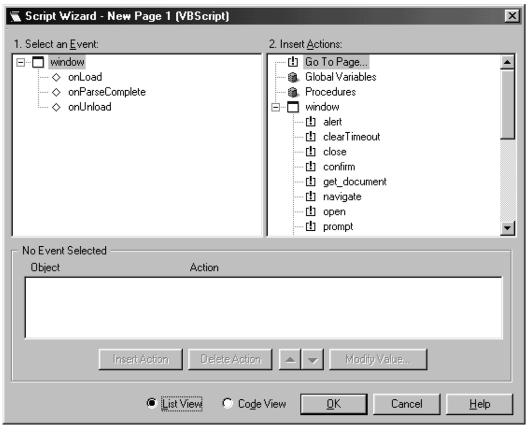
```
<SCRIPT Language = "VBScript">
<! - -
Sub ImageClicked( )
  Window.Navigate ("http://www.newsite.com/")
End Sub
- ->

</SCRIPT>
<IMG SRC="myimage.gif" OnClick="ImageClicked">
```

דרך פשוטה שתאפשר לגלות באילו אירועים תוכל להשתמש בעת עבודה עם אובייקט מסוים, היא על ידי שימוש ב- Script Wizard של תוכנת FrontPage. שלב אובייקט מסוג זה במסמך FrontPage והצג את כלי Script Wizard שמוצג בתרשים 30.7.

בדוגמת הקוד הקודמת התייחסנו לשיטה Navigate של אובייקט Internet Explorer .Window שיטה זו מורה ל- Internet Explorer לפתוח כתובת URL נתונה. Document.write היא פונקציה מועילה נוספת, שמציגה תגי HTML בחלון הדפדפן:

```
<SCRIPT Language = "VBScript">
<! - -
document.write "The current date and time is" & now
- - >
</SCRIPT>
```



תרשים 30.7: FrontPage Script Wizard מציג אירועים במבנה היררכי

רשימה מפורטת של האובייקטים והשיטות הנתמכים על ידי Internet Explorer תוכל למצוא באתר Microsoft.

טפסים

טפסים (Forms) הם אלמנטים של HTML המשמשים לשליחת נתונים חזרה לשרת. קטע הקוד הבא יגדיר טופס אשר יכיל שתי תיבות טקסט:

```
<FORM Action="formproc.asp" METHOD="POST" NAME="frmTest">
Please enter your name and E-Mail address below:<BR>
Name: <INPUT Type=Text Size=40 name=txtName> <BR>
Email: <INPUT Type=Text Size=30 name=txtEmail> <BR>
<INPUT Type=Submit Name=cmdSend Value="Send Values to Server">
</FORM>
```

לחצן Submit מורה לדפדפן להעביר את כל הנתונים המאוחסנים בשדות <INPUT> שתחומים בין תגי <FORM> לשרת. לאחר ביצוע ההעברה, יישום השרת (שבדוגמה המתוארת כאן הוא קובץ Active Server Page בשם formproc.asp) יוכל לגשת לכל פריט שייכלל בטופס, בנפרד, ולאחסן אותו במסד הנתונים. אך ייתכן שלפני שתעביר את הנתונים לשרת, תרצה להיעזר בקוד VBScript כדי לבצע על הנתונים בדיקות תקינות או כדי לבצע שינויים כלשהם. תוכנית 30.3 מכילה דוגמה לשימוש בשפת VBScript לביצוע בדיקות תקינות על הנתונים שבטופס.

תוכנית 30.3: VBSFORMS.HTM ביצוע בדיקות תקינות ושינויים בנתונים שבטופס.

```
<HTML>
<HEAD><TITLE>Simple example of Forms</TITLE></HEAD>
<SCRIPT Language = "VBScript">
<!--
Sub submitform()

 Dim nPos
 Dim sUserType
 Dim f

 Set f = Document.frmMain

 'Make sure name is not blank
 If Trim(f.txtName.Value) = "" Then
      Msgbox "Please enter your name and try again!"
      Exit Sub
 End if

 'Validate E-Mail Address Format
 nPos = Instr(f.txtEmail.Value,"@")
 If nPos = 0 Then
      Msgbox "Please use the format user@server.domain"
      Exit Sub
 End If
  'Classify user as business or other
```

```
  nPos = Instr(LCase(f.txtEmail.Value),".com")
  If nPos <> 0 Then '.com = commercial domain
        sUserType = "BUSINESS"
  Else
        sUserType = "OTHER"
  End If

  'Put user type in hidden form field
  f.txtUserType.Value = sUserType

Msgbox "Form is ready to submit!"
Exit Sub

  'Submit the form
  f.Method = "POST"
  f.Action = "formproc.asp"
  f.submit

End Sub
-->
</SCRIPT>

<BODY>

 Please fill out all fields below:<BR><BR>

<FORM NAME=frmMain>

 Name: <INPUT Type=Text maxlength=20 size=20 name=txtName>  <BR>
 Email: <INPUT Type=Text maxlength=30 size=30 name=txtEmail> <BR>

 <INPUT Type=Hidden name=txtUserType>

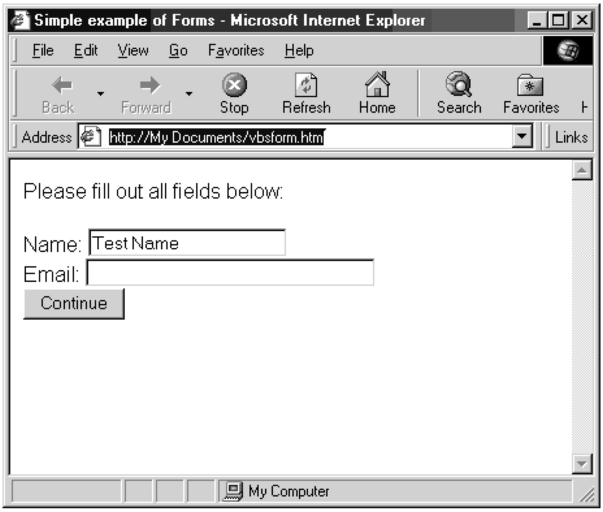
 <INPUT Type=Button OnClick=submitform Value="Continue">
</FORM>

</BODY>
</HTML>
```

אזהרה:

זכור שלא ניתן להשתמש בשפת VBScript בעת עבודה עם דפדפן Netscape, על-כן ייתכן שכדאי שתבצע את בדיקות תקינות הנתונים בשרת.

בתוכנית 30.3 ניתן לראות שרוב המטלות שקשורות בטיפול בטופס מבוצעות באמצעות VBScript במקום באמצעות HTML. לדוגמה, הפרמטרים ACTION ו-METHOD הושמטו משורת ההצהרה של הטופס. לא כללנו בטופס לחצן Submit מפני שפעולת העברת הנתונים לשרת בוצעה באמצעות שימוש בשפת VBScript. שים לב גם לשדה שהוגדר Hidden (מוסתר) שבו מאוחסן ערך שלא הוזן על ידי המשתמש. בתרשים 30.8 תוכל לראות דוגמה לשימוש בטופס שהוגדר בתוכנית 30.3.



תרשים 30.8: VBScript מבצעת בדיקות תקינות נתונים לפני שהם מועברים לשרת

פקדי ActiveX

בנוסף לשימוש בשיטה CreateObject ליצירת אובייקטים, תוכל להסתייע בתג <OBJECT> לשם הטבעת אובייקטים בדפי Web. תג זה מאפשר להציב פקדי ActiveX בדפי Web. אחר תוכל להסתייע בקוד VBScript כדי לגשת לאובייקטים, כמו בדוגמה לשימוש בתיבת רשימה, שמובאת להלן:

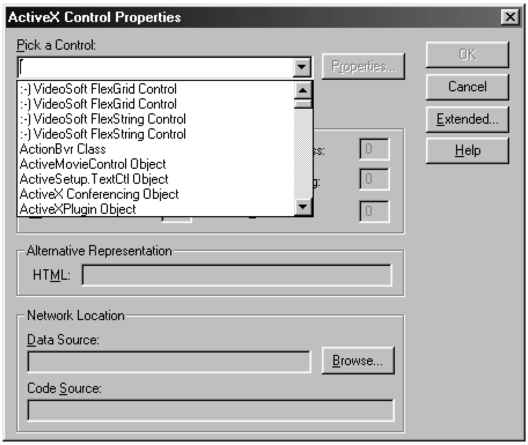
```
<HTML>
<BODY>
<OBJECT id="lstmain" classid="clsid:8BD21D20-EC42-11CE-9E0D-00AA006002F3"
width=152 height=164>
<PARAM name=ScrollBars value=3>
<PARAM name=DisplayStyle value=2>
</OBJECT>
<SCRIPT Language = "VBScript">
<!- -
  Dim i
  For i = 1 to 10
    lstMain.AddItem "Item " & i
Next

Sub lstMain_Click()
   Msgbox "You Clicked " & lstMain.List(lstMain.ListIndex)
End Sub
 - - >
</SCRIPT>
</BODY>
</HTML>
```

להלן פירוט מרכיבי תג <OBJECT>:

- ID - מציין את השם שישמש את VBScript לצורך פנייה לאובייקט. מקביל למאפיין Name שמוגדר ב- Visual Basic.
- CLASSID - משמש כמזהה ייחודי לזיהוי האובייקט המאוחסן ברישום המערכת של Windows.
- CODEBASE - מכיל את כתובת URL של קובץ OCX או CAB שממנו ניתן להוריד את האובייקט. בדוגמה לא נעשה שימוש בפרמטר CODEBASE מפני שפקד ListBox הוא פקד המובנה ב-Internet Explorer, כך שלא נדרש להגדיר את הפרמטר CODEBASE בפנייה אליו.
- height ו-width - מגדירים את גודל האובייקט.
- תגי PARAM - מגדירים את אופן פעולת האובייקט, באופן דומה לחלון המאפיינים ב- Visual Basic.

הדרך הטובה ביותר לשלב פקדי ActiveX בדפי Web היא על ידי שימוש בתוכנת FrontPage. כאשר לאחר סיום פעולת ההוספה, העתק את קוד HTML שנוצר על ידי FrontPage לדף Web. בתרשים 30.9 ניתן לראות את רשימת פקדי ActiveX שנתמכים על ידי FrontPage.



תרשים 30.9: Class ID ניתן לאתר באמצעות FrontPage. פקד ששמו מתחיל ב- Microsoft Forms הוא מובנה של Internet Explorer ואין צורך להורידו מה-Web

בפרק 14 למדת כיצד ליצור פקדי ActiveX משלך, שאותם תוכל לשלב בדפי Web. שימוש בפקדים שתיצור מציע מספר יתרונות על שימוש ב-VBScript או ב-HTML, שביניהם:

- שימוש בפקדים אלה מאפשר להשתמש במלוא מיגוון היכולות של Visual Basic.
- משתמשים לא יוכלו להציג את הקוד שלך בקלות, כפי שיתאפשר להם בעת שימוש ב-Scripts.

Windows Scripting Host

בנוסף למיגוון האופנים בהם ניתן להשתמש ב-VBScript ב-Web, החלה Microsoft להציע את VBScript כתחליף לקבצי אצווה. לאחר שתתקין את Windows Scripting Host הניתן להורדה חינם מאתר http://www.microsoft.com/scripting, תוכל להריץ Scripts של VBScript משורת הפקודה של DOS או מסייר Windows. את קבצי קוד VBScript תיצור בעזרת עורך טקסט ותאחסן תוך שימוש בסיומת VBS.

הפעלת Script

תוכנת Windows Scripting Host תתקין במחשב שני יישומים מארחים, שיהיו מסוגלים להפעיל קוד VBScript: CSCRIPT.EXE המיועד לפעול משורת הפקודה של DOS ו-WSCRIPT.EXE המיועד לעבודה בסביבת Windows.

כדי להמחיש את ההבדל בין שני היישומים, ניצור קובץ VBS שיכיל פנייה לשיטה WScript.Echo, כמו בדוגמה הבאה:

```
WScript.Echo "Hello, World!"
```

הפעלת Script זה תוך שימוש בשפת CScript גורמת להצגת ההודעה Hello, World בחלון הרצה של DOS ואילו הפעלתו באמצעות WScript מציגה את ההודעה בתיבת הודעה בסביבת Windows.

הדרך הפשוטה ביותר להפעיל Script היא על ידי העברת שמו כפרמטר לאחד משני קבצי הביצוע שצוינו קודם לכן, כמו בדוגמה הבאה:

```
CScript.exe d:\scripts\test.vbs
```

תוכל גם ללחוץ לחיצה כפולה על קובץ Script בחלון Windows Explorer. Scripting Host תומך בכמה פרמטרים המועברים משורת הפקודה, אשר מפורטים בטבלה 30.5. בעת שימוש בפרמטרים אלה יש להקליד שני לוכסנים (//) לפני שם הפרמטר:

טבלה 30.5: פרמטרי Scripting Host

פרמטרים	תיאור
//B //I	מאפשרים לבחור בין הפעלה באצווה לבין הפעלה אינטראקטיבית. בעת הפעלה באצווה לא מוצגים סמנים מנחים כלשהם
//logo //nologo	שולטים על הצגת פרטי זכויות יוצרים
//T:nn	מגדיר משך זמן נתון בשניות שלאחריו יתעורר **פסק זמן** בפעולת Script. הפעולה תופסק לאחר שיחלוף הזמן שהוגדר
//H:CScript	מקשר Scripts ליישום המארח CScript
//H:WScript	מקשר Scripts עם היישום המארח WScript
//S	שומר את האפשרויות הפעילות (תקף רק בסביבת Windows)

את האפשרויות המפורטות בטבלה 30.5 תוכל להגדיר גם בקבצי WSH. קובץ WSH דומה לקובץ PIF, אך משמש לעבודה עם קבצי אצווה. הוא מכיל את האפשרויות במבנה של קובץ INI :

```
[ScriptFile]
Path=D:\Scripts\test.vbs

[Options]
Timeout=10
DisplayLogo=0
BatchMode=0
```

את הגדרות הארגומנטים שיועברו ל-Script עצמו, יש לפרט לאחר הגדרת האפשרויות ושם ה-Script. את הערכים שיועברו לארגומנטים תוכל לאחזר על ידי שימוש באוסף Arguments של אובייקט WScript :

```
Dim nCount
nCount = WScript.Arguments.Count
WScript.Echo "There are " & nCount & " arguments."
For i = 0 to nCount - 1
 Wscript.Echo Wscript.Arguments(i)
Next
```

שיטות ואובייקטים שימושיים

מידת השימושיות של ספריית אובייקטים תלויה בסוגי האובייקטים והשיטות שהיא מציעה למפתחים. משום ש- Windows Scripting Host נועד להוות תחליף לקבצי אצווה, הוא תומך בביצוע פעולות רבות הקשורות בשימוש במערכת ההפעלה, ביניהן :

- יצירת קיצורי דרך על שולחן העבודה.
- מיפוי כונני רשת.
- גישה לרישום המערכת.
- עבודה עם אובייקטים התומכים במיכון (Automation).
- עבודה עם חשבונות משתמש Windows NT 5.0.

באתר Microsoft תוכל למצוא את התיעוד המלא של Windows Scripting Host ודוגמאות קוד שמבצעות את כל סוגי הפעולות שפורטו כאן. קיימים שלושה סוגי אובייקטים עיקריים שמאפשרים לבצע פעולות כאלו. קודם לכן הצגנו דוגמה לשימוש באובייקט WScript. ניתן לגשת לאובייקט WScript ישירות מקוד VBScript.

הערה:

את Windows Scripting Host אפשר להוריד מאתר Microsoft העוסק בנושא Scripting Technologies שכתובתו http://www.microsoft.com/scripting.

לשני סוגי האובייקטים האחרים, Shell ו-Network יש לפנות מתוך אובייקט WScript על ידי שימוש בשיטה CreateObject. לדוגמה, קטע הקוד הבא ישתמש בשיטה Run של אובייקט Shell, כדי להפעיל תוכנית ולהמתין לסיום פעולתה:

```
Dim WshShell
Set WshShell = WScript.CreateObject("WScript.shell")
'First parameter is Window style, Second is Wait
WsShell.Run "notepad", 1, True
MsgBox "Done!"
```

לאחר שתתנסה בשימוש ב-Windows Scripting Host תיווכח שהוא כלי רב עוצמה למיכון פעולות בסביבת Windows.

מכאן...

פרק זה הציג דרכים בהן תוכל ליישם מיומנויות ב- Visual Basic במסגרת ה-Web. מידע נוסף הקשור בנושאים אלה תוכל למצוא בפרקים הבאים:

- מידע אודות שימוש בשפת VBScript בשרתי Web להפעלת דפי Web תוכל למצוא בנספח 4 **דפי שרת פעילים**.
- מידע נוסף אודות הסבת יישומים עצמאיים שנכתבו ב- Visual Basic לעבודה ב-Web, תוכל למצוא בפרק 32 **Visual Basic ושימושים נוספים באינטרנט**.

31

מסמכי ActiveX

מה בפרק?

- הבנת מסמכי ActiveX
- יצירת מסמך ActiveX
- חקירת אובייקט UserDocument
- אובייקט HyperLink במסמכים
- אשף ActiveX Document Migration
- יצירת מסמך מורכב

בוודאי ידוע לך שב-Web מתרחשים כיום תהליכים רבים. התהליך העיקרי המתרחש הוא ש- World Wide Web הולך וצובר פופולריות עצומה. לעיתים נדמה לך שלכל אדם ולכל ארגון שבהם אתה נתקל יש אתר Web - החל בגופים מסחריים דרך ארגוני צדקה וכלה בשכנים שלך. דברים הקשורים ב-Web הפכו למצרך המצוי בכל מקום ולמרות שבעבר ניתן היה להסתפק בדפי Web סטטיים, כיום יותר ויותר אנשים נוהגים לכלול תוכן דינמי באתרים שלהם. דפי Web אינטראקטיביים הקיימים כיום אינם מסתפקים בהצגת מידע קבוע. כיום לא נדיר להיתקל בדפי Web שמתפקדים כיישומים לכל דבר, ולכן גם מוצעים כלים שונים שמסייעים ביצירת דפים כאלה.

כאשר תשקול לראשונה לשלב תוכן דינמי בדפי Web, התהליך עלול להיראות לך מרתיע עקב ריבוי האפשרויות שביניהן תידרש לבחור לביצוע פעולות אלו. מרבית התהיות הרלוונטיות שאני שומע כיום אינן בנוסח "האם אוכל בכלל לבצע את המשימה?" אלא בנוסח "מהי הדרך הטובה ביותר לבצע משימה זו?". כאשר תנסה לחשוב על דרכים שבהן תוכל לפתח בסביבת Web תיתקל בעשרות מושגים רלוונטיים כגון Perl, CGI, Active Server Pages, VBScript, Java, ועוד. **מסמכי ActiveX** (ActiveX Documents) אינם בהכרח הכלי הטוב ביותר לשימוש בכל המקרים, אך הם מהווים אמצעי שהופך את ה-Web לנגיש למפתחי Visual Basic.

הבנת מסמכי ActiveX

משום שמסמכי ActiveX הם כלי נוח מאוד לשימוש ב- Web, נפתח את פרק זה ב"קורס רענון" קצר. סביר להניח שכבר רכשת ידע מספיק על דפי Web שמאפשר לך להבין שדפים אלה הם למעשה קבצי מסמך. קבצי Web דומים לקבצי Word אך הם כתובים בפורמט מיוחד הנקרא HTML (Hypertext Markup Language). באותו אופן בו Word משמש ככלי להצגת קבצי DOC כך דפדפן Web (כגון Netscape או Internet Explorer) משמש להצגת קבצי HTML. לכל קובץ HTML המאוחסן ב-Web יש כתובת המכונה כתובת URL (Uniform Resource Locator - מאתר משאבים אחיד) המשמשת לאיתור הקובץ.

Web החל דרכו כאוסף מסמכים שהיו מקושרים אחד לשני. אך המסמכים הסטטיים שהיו קיימים אז לא אפשרו לממש את רמות האינטראקטיביות המוצעות כיום. קיימים שני גורמים הקשורים בעבודה ב-Web שאפשרו שינוי מצב זה:

- שינוי האופן בו דפי טקסט מאוחזרים.
- שיפורים שחלו בדפדפנים.

פרוטוקול HTTP (Hyper Text Transfer Protocol) הוא האמצעי המקשר בין הדפדפן שלך ושרת Web. הדפדפן מעביר בקשה לקבלת כתובת URL מסוימת, ומציג את זרם נתוני HTML המוחזרים אליו.

שים לב שהשתמשתי במונח **בקשה** (Request). מונח זה מבוסס על תפיסה חשובה. העברת בקשה לקבלת קובץ מסוים שונה מאוד מפתיחת קובץ המאוחסן בדיסק הקשיח שלך, והצורך להשתמש בפרוטוקול תקשורת נובע משוני זה. ננסה לדמות

תהליך זה למצב בו אתה מבקש מחבר להשאיל לך מסמך מסוים, על ידי כך שהתה מעביר לו את הבקשה באמצעות דואר אלקטרוני. בשונה ממצב בו אתה פותח קובץ המאוחסן בדיסק שלך, העברת בקשה לשאילת המסמך מאפשרת לגורם המשאיל לערוך את המסמך לפני שהוא יעביר אותו אליך או אפילו להעביר אליך מסמך שונה לחלוטין. כך תוכל לאחסן בשרת שלך לוגיקה שתשנה את המסמך שיוחזר בהתאם למערכת קריטריונים מסוימת, שעשויה להתייחס גם לנתונים שיתקבלו מהמשתמש.

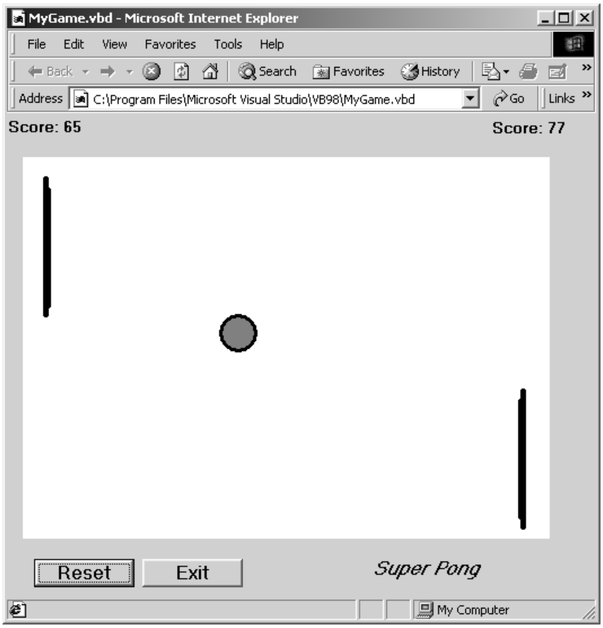
גם זרם נתוני HTML המתקבל בדפדפן השתנה לא מעט מאז ראשית ימי ה-Web. בנוסף לטקסט מעוצב ולגרפיקה, דפי Web יכולים כיום להכיל קוד Script ויישומוני Java. שיפורים אלה התאפשרו משום שדפדפני Web הפכו מורכבים יותר כדי לאפשר תמיכה בכל האובייקטים המשופרים שמוטבעים בדפי Web. מסמכי ActiveX שיידונו בפרק זה הם יישומים אשר Internet Explorer מעביר לעמדת העבודה של המשתמש.

מהו מסמך ActiveX

בהגדרה הפשטנית ביותר של המושג, מסמך ActiveX הוא יישום הפועל בתוך יישום מכיל (Container), כגון Internet Explorer במקום לפעול כתוכנית עצמאית. בתרשים 31.1 מוצגת דוגמה ליישום כזה.

הערה:

ניתן להשתמש במסמך ActiveX ב- Internet Explorer, אך תידרש להתקין במחשב המשתמש את כל קבצי Run Time DLL הדרושים לשימוש במסמך.



תרשים 31.1: מסמך ActiveX פשוט המוצג כאן פועל ב- Internet Explorer

מקור המילה "מסמך" (Document) שמופיע בביטוי **מסמך ActiveX** הוא מהקבלה למסמכי עיבוד תמלילים או מסמכי גליונות אלקטרוניים. קבצים אלה מכילים נתונים, אך יש להשתמש בתוכנית כדי שניתן יהיה להציגם או לערוך אותם. לדוגמה, תוכל ליצור מסמך על ידי שימוש בתוכנת Word ולאחסן אותו כקובץ. אם תעביר קובץ זה למשתמש אחר, הוא לא יוכל לעשות איתו כלום, אלא אם כן תהיה ברשותו התוכנה המתאימה. אופן פעולת מסמך ActiveX דומה. תוכל ליצור מסמך ActiveX ולאחסן אותו בקובץ. אם תעביר את הקובץ למישהו אחר, הוא יידרש להשתמש בתוכנית התומכת במסמכי ActiveX כדי שיוכל להשתמש בקובץ.

למרבה המזל מוצעים כיום מספר יישומים מכילים אשר תומכים במסמכי ActiveX, ביניהם Internet Explorer, אוגדן (Binder) של Office 97 וסביבת הפיתוח המשולבת (Integrated Development Environment - IDE) של Visual Basic. משתמשים יוכלו להפעיל את מסמכי ActiveX שתפתח ביישומים אלה.

יתרונות השימוש במסמכי ActiveX

הסיבה העיקרית שבעטיה כדאי להשתמש במסמכי ActiveX ב- Visual Basic, היא כדי ליצור יישומים בעלי יכולות לעבודה ב-Web. יצירת מסמכי ActiveX מציעה מספר יתרונות שאין לשיטות אחרות המשמשות ליצירת מסמכים לעבודה ב-Web. חלק מהיתרונות יפורטו ברשימה הבאה:

- לא תידרש ללמוד שפת תכנות נוספת כדי ליצור מסמכים כאלה. תוכל להסתפק בידע ב- Visual Basic.
- תוכל לעצב יישומי Web על ידי שימוש בסביבת העיצוב של Visual Basic. בהשוואה לשיטות לכתיבה וניסוי קוד המיושמות על ידי שפות אחרות, שיטה זו תקל את העבודה.
- אופן עבודה זה יאפשר גישה לכלי ניפוי שגיאות של Visual Basic הכלי ייסייע לך לנפות את הקוד ולפתור בעיות.
- אובייקט Hyperlink יקל את השימוש בדפדפן לניווט בדפי Web. הדפים עשויים להיות מסמכי ActiveX או כל סוג אחר של מסמכי Web.

מהנאמר לעיל הבנו כי Visual Basic מסייעת ביצירת מסמכי ActiveX, אך מדוע שתרצה בכלל ליצור מסמכים כאלה? מדוע שלא תיצור יישומים רגילים? התשובה היא בשתי מילים "בגלל ה-Web". יצירת מסמכי ActiveX חוסכת את הטרחה הכרוכה ביישום שיטות מיושנות יותר להפצת תוכנות. אם התוכנית שלך היא קובץ EXE סטנדרטי, תצטרך להפיצה למשתמשים על ידי משלוח דיסקטים של גרסת ההתקנה. מאידך, בעת שימוש במסמכי ActiveX תוכל לאחסן עותקים של קבצי ההפצה על שרת Web, באופן שיאפשר למשתמשים להורידם מייד בכניסתם לאתר שלך. קוד שיוטמע בדף Web יורה לשרת להוריד קבצי Cabinet (קובץ CAB - קובץ דחוס) שיכילו את היישום שלך למחשב המשתמש, כשהם מלווים בכל הרכיבים הדרושים לשימוש ביישום. זוהי שיטת הפצה המבוססת על שימוש ב-Web, המקלה על תחזוקת הקוד ומוודאת אחידות בין הגרסאות שהופצו.

מובן שלשימוש במסמכי ActiveX ב-Web יש גם חסרונות מסוימים. להלן כמה מביניהם:

- נכון לכתיבת שורות אלו, רק Internet Explorer תומך ב-ActiveX. על-כן, אם אתה מייעד את היישום לשימוש כללי ב-Web (ולא רק ברשת פנימית בחברה מסוימת). כדאי שתבנה אותו תוך יישום גישה המיושמת בשרת, כמו למשל שימוש בדפי שרת פעילים (Active Server Pages).
- Internet Explorer אומנם מטפל במה שקשור להורדת היישום למחשב המשתמש, אך המשתמש עדיין נדרש להתקין את היישום על המחשב שלו, כשהוא מלווה בכל קבצי העזר הדרושים. בכל פעם שתיצור גירסה חדשה של מסמך ActiveX, המשתמשים יידרשו לעדכן את העותקים שלהם, פעולה שעשויה להימשך זמן רב בחיבורים איטיים ל-Web.

יצירת מסמך ActiveX

יצירת מסמך ActiveX ב- Visual Basic דומה ליצירת יישום רגיל. בפרק זה תיצור שוב את התוכנית לחישובי החזרי הלוואות שיצרת בפרק 2, אך הפעם תיצור אותה כמסמך ActiveX. צור את המסמך על ידי ביצוע סדרת הפעולות שלהלן:

1. התחל פרויקט חדש מסוג ActiveX Document.
2. צור את ממשק המשתמש של היישום.
3. כתוב את הקוד שיבצע את הפעולות הנדרשות.
4. נסה את היישום ונפה ממנו שגיאות.
5. היעזר בכלי Package and Deployment Wizard (אשף האריזה וההפצה) כדי ליצור גרסת התקנה שתופץ ב-Web.

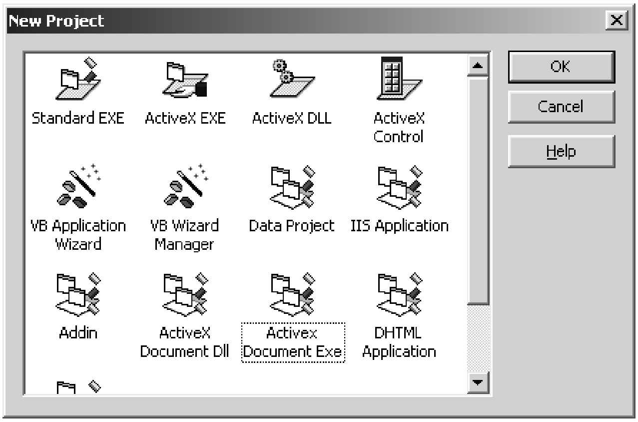
כל אחד מהצעדים שפורטו כאן מורכב ממספר פעולות, אך התהליך כבר מוכר לך. נדגים את יצירת המסמך על ידי יצירת התוכנית לחישובי החזרי הלוואות, כמו זו שיצרת בפרק 2. השימוש באותו יישום ימחיש לך את נקודות הדמיון וההבדלים הקיימים בין יצירת מסמך ActiveX לבין יצירת יישום רגיל הכתוב ב- Visual Basic.

הערה:

בפירוט הצעדים התייחסנו למושג יישום (Application). משום שמסמך ActiveX הוא יישום אינטראקטיבי ולא קובץ סטטי כמו למשל קובץ מסמך עיבוד תמלילים, נהוג להשתמש במושג יישום לצורך התייחסות אליו.

פרויקט מסמך ActiveX

הצעד הראשון ביצירת מסמך ActiveX הוא התחלת פרויקט חדש. התחלת פרויקט חדש נעשית על ידי בחירה בפקודה **New Project** מתפריט **File**. תוצג תיבת הדו-שיח New Project (ראה תרשים 31.2). בחר את הסמל ActiveX Document EXE על ידי לחיצה כפולה עליו. בחירה זו תיצור פרויקט חדש שיכיל אובייקט User Document יחיד בשם UserDocument1. לחץ לחיצה כפולה על האובייקט בחלון Project Explorer כדי להציג מסך עיצוב ריק כמו זה הנראה בתרשים 31.3.



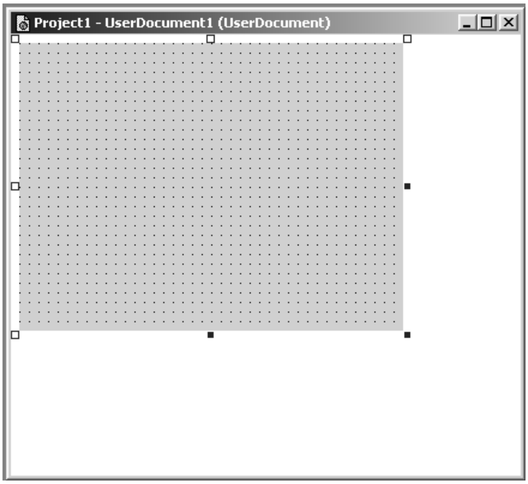
תרשים 31.2: תיבת הדו-שיח New Project, שאותה ניתן להציג על ידי הקשה Ctrl+N משמשת להגדרת התצורה ההתחלתית של הפרויקט

הערה:

אם אובייקט UserDocument אינו מוצג אוטומטית, לחץ לחיצה כפולה על אובייקט UserDocument שבחלון Project.

שים לב שחלון עיצוב אובייקט UserDocument דומה מאוד לחלון עיצוב הטפסים, אך הוא אינו כולל את הגבול שבחלון עיצוב הטפסים. חלון זה דומה גם לאובייקט UserControl שמשמש ליצירת פקדי ActiveX. בחלון זה תיצור את ממשק המשתמש של פקד ActiveX, באותו אופן שבו שיצרת את ממשק המשתמש של טפסים (מידע נוסף אודות יצירת פקדי ActiveX תוכל למצוא בפרק 14).

לאחר שתיצור את הפרויקט, שנה את מאפייניו ומאפייני אובייקט UserDocument לפי טבלה 31.1. לגישה למאפייני הפרויקט, בחר את הפקודה **Properties** מהתפריט **Project**. תוכל להגיע למאפייני אובייקט UserDocument גם מחלון המאפיינים (על ידי בחירה בפקודה **Properties Window** מתפריט **View**). לאחר שתגדיר את המאפיינים, שמור את קבצי הפרויקט על ידי לחיצה על לחצן שמירה שבסרגל הכלים. ותן שמות לקבצים החדשים.



תרשים 31.3: מסמכי ActiveX פועלים בתוך יישומים מכילים ולכן הם חסרים שורות כותרת, גבולות ואלמנטים אחרים הקיימים בחלונות עצמאיים

טבלה 31.1: מאפייני פרויקט ואובייקט UserDocument

מאפיין	הגדרה
Project Type	ActiveX EXE
Name Project	ActXCalc
טProject Description	ActiveX Document Loan Calculator
UserDocument Name	CalcDoc

שמות קבצי המסמכים

את הקוד במסמכי ActiveX ניתן לשמור בקובץ טקסט, באופן דומה לשמירת טפסים. תיאור אובייקט UserDocument והפקדים שבו מאוחסן יחד עם הקוד המסמך בקובץ בעל סיומת DOB שמבנהו דומה למבנה קובץ FRM שבו מאוחסנים טפסים. אם ממשק הטופס מכיל אלמנטים גרפיים כלשהם, הם מאוחסנים בקובץ DOX, הדומה לקובץ FRX שמשמש לאחסון האלמנטים הגרפיים של טפסים. כאשר תהדר מסמך ActiveX תיצור קובץ EXE או קובץ DLL שילווה בקובץ VBD. קובץ VBD הוא הקובץ שאליו ניגש Internet Explorer. קובץ VBD מכיל את ה״מסמך״ בדומה לקובץ DOC.

יצירת ממשק המסמך

אתה יוצר את ממשק מסמך ActiveX על ידי שירטוט פקדים על אובייקט UserDocunent באותו אופן בו משרטטים פקדים על טופס רגיל. במסמכי ActiveX ניתן לשלב כמעט את כל סוגי הפקדים המוצעים על ידי Visual Basic. היחיד שלא ניתן להשתמש בו הוא פקד OLE Container. הגבלה נוספת היא שמסמכי ActiveX אינם יכולים להכיל אובייקטים מוטבעים כדוגמת מסמכי Excel או Word.



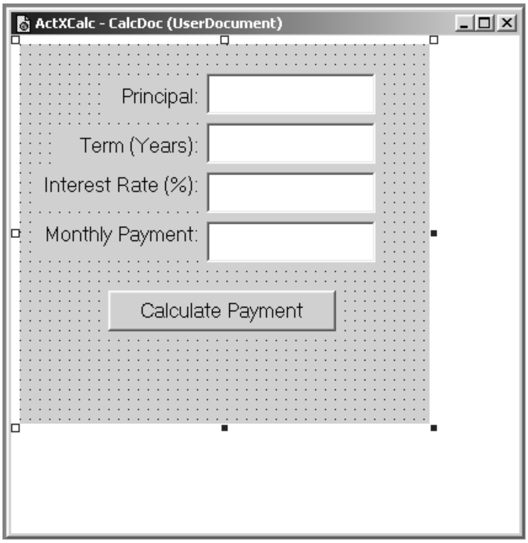
אזהרה:

אם תשלב פקדים מותאמים אישית במסמכי ActiveX, בעת הפצת המסמכים תידרש להתייחס לנושא רישוי השימוש בפקדים.

לצורך יצירת ממשק יישום הדוגמה תידרש להוסיף לאובייקט UserDocument ארבעה פקדי Label, ארבעה פקדי TextBox ופקד ControlButton.

הגדר את מאפייני Name ו-Caption של פקדי Label כלהלן:

Name	Caption
lblPrincipal	Principal:
lblTerm	Term (Years):
lblInterest	Interest Rate (%):
lblPayment	Monthly Payment:



תרשים 31.4: המסמך Calculator ידמה לתוכנית LoanCalc שאותה יצרת קודם לכן

תצטרך להציב מחרוזות ריקות במאפיין Text של ארבעת פקדי TextBox, כדי שתיבות הטקסט תוצגנה כשהן ריקות. קרא לתיבות הטקסט: txtInterest, txtPayment, txtPrincipal, ו-txtTerm.

הגדר את מאפיין Name של פקד CommandButton כ-cmdCalculate ואת מאפיין Caption של הלחצן כ-Calculate Payment.

לאחר שתוסיף את הפקדים אובייקט UserDocument ייראה כמוצג בתרשים 31.4.

הוספת קוד למסמך

לאחר שתיצור ממשק משתמש עבור המסמך, על ידי שימוש בפקדי Visual Basic תהיה מוכן לכתוב את הקוד. כתיבת קוד לפקדים שבמסמך ActiveX דומה לכתיבת קוד לפקדים בטופס רגיל. תוכל להציג את חלון הקוד באמצעות לחיצה כפולה על פקד, או על ידי לחיצה על לחצן View Code שבחלון הפרויקט. בעת בניית יישום הדוגמה הזן את הקוד שבתוכנית 31.1 באירוע Click של לחצן הפקודה.

תוכנית 31.1: MYCALC.VBP - הוספת קוד לביצוע חישובים לשגרת האירוע Click

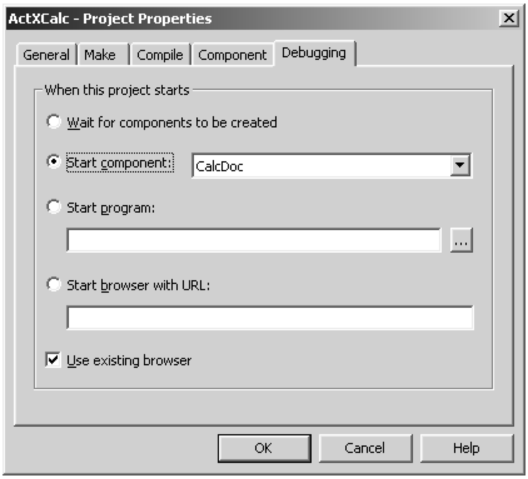
```
Private Sub cmdCalculate_Click( )
On Error Resume Next
Dim m_Principal As Single, M_Interest As Single
  Dim m_Payment As Single, m_Term As Integer
  Dim m_fctr As Single
  m_Principal = Val(txtPrincipal.Text)
  m_Interest = Val(txtInterest.Text) /1200
  m_term = Val(txtTerm.Text)
  m_fctr(1 + m_Interest) ^ (m_Term * 12)
  m_Payment = m_Interest * m_fctr * m_Principal / (m_fctr - 1)
  txtPayment.Text = Format(m_Payment, "Fixed")
End Sub
```

אומנם הקוד שבתוכנית 31.1 קיים באובייקט UserDocument, אך הוא פועל בדיוק כמו קוד שבטופס. הקוד מאחזר תחילה את הערכים שהמשתמש הזין בתיבות טקסט ומאחסן אותם במשתנים מקומיים. לאחר מכן מתבצע חישוב סכום התשלום, שתוצאתו מוצגת בתיבת הטקסט txtPayment.

ניסוי המסמך

לאחר שתזין קוד ותשמור את המסמך, תהיה מוכן לנסותו. הפעולות הדרושות שונות מעט מאלו הדרושות לניסוי יישום רגיל, מפני שמסמך ActiveX פועל כשהוא מוכל ביישום אחר. כדי להגדיר איזה יישום יופעל על ידי המסמך, בעת בניית הפרויקט הצג את הכרטיסיה Debugging שבתיבת הדו-שיח Properties. תוצגנה מספר אפשרויות, שאותן ניתן לראות בתרשים 31.5.

אפשרות ברירת המחדל בכרטיסיה Debugging, היא Start Component, והיא מאפשרת לקוד הרכיב לקבוע את האופן בו יופעל. אם מותקן Internet Explorer ותתחיל פרויקט חדש של יצירת מסמך ActiveX, אפשרות ברירת המחדל תגרום לכך שהמסמך ייטען ל-Internet Explorer. נסה כעת להפעיל את פרויקט הדוגמה כדי להיווכח בכך. הקש F5 או לחץ על לחצן Start כדי להציג את גרסת Web של תוכנית חישוב התשלומים, שנראית בתרשים 31.6.



תרשים 31.5: כרטיסיית Debugging שבתיבת הדו-שיח Properties תאפשר לקבוע האם המסמך יופעל בדפדפן או ביישום אחר

אם תרצה לפתוח את מסמך ActiveX באופן ידני, בחר את האפשרות **Wait for Components to Be Created** שבכרטיסיה Debugging. בחירה באפשרות זו תגרום ל-Visual Basic להפעיל את הפרויקט בלבד, אתה תידרש לפתוח את המסמך באמצעות Internet Explorer. כדי לפתוח מסמך ActiveX בצע את הפעולות הבאות:

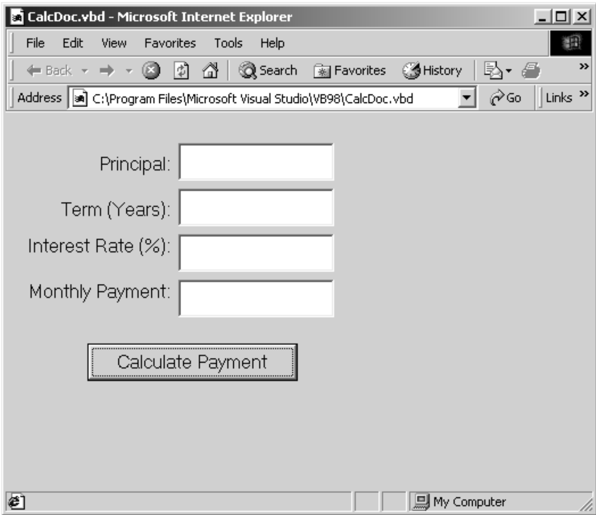
1. הפעל את המסמך על ידי הקשת F5 או על ידי לחיצה על לחצן Start שבסרגל הכלים (שים לב ש- Visual Basic לא הציגה את ממשק המשתמש של התוכנית).
2. מזער את סביבת הפיתוח של Visual Basic והפעל את Internet Explorer.
3. בחר את הפקודה **Open** מתפריט **File** של Internet Explorer. הזן את שם הקובץ שאותו אתה מעוניין לפתוח בתיבת הדו-שיח שתוצג.

4. הגדר את הנתיב ואת שם קובץ המסמך. השם שתגדיר מורכב מהערך שבמאפיין Name של אובייקט UserDocument וסיומת VBD. אם תפעיל את המסמך מתוך Visual Basic, המסמך יהיה מאוחסן בתיקיית Visual Basic. להתקנת Typical הנתיב הוא C:\Program Files\Microsoft Visual Studio\VB98.

5. לחץ על לחצן OK שבתיבת הדו-שיח Open כדי לטעון את המסמך. בתרשים 31.6 ניתן לראות את המסמך CalcDoc כשהוא פועל ב- Internet Explorer.

הערה:

אם תשתמש בלחצן Browse שתיבת הדו-שיח Open לחיפוש הקובץ, זכור לבחור את האפשרות All Files בתיבת הרשימה הנפתחת Files of Type.



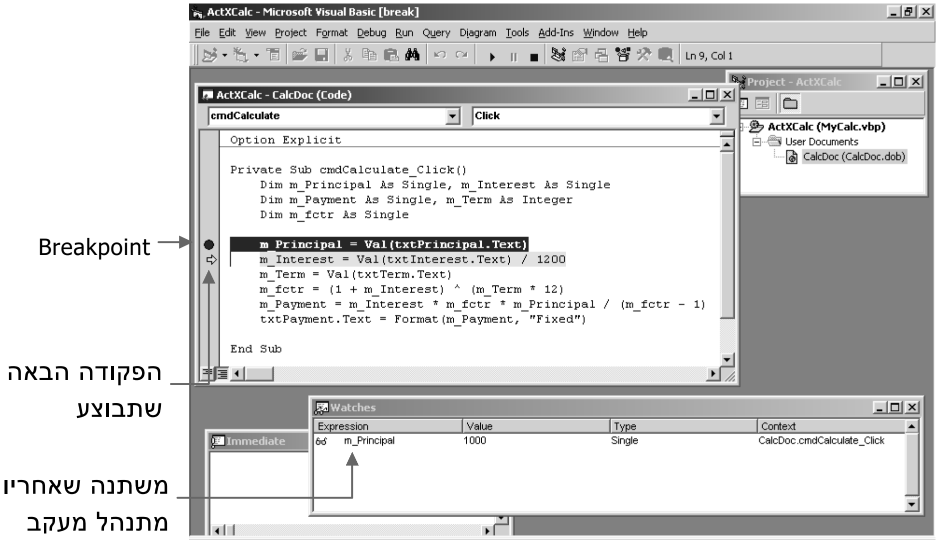
תרשים 31.6: Internet Explorer יכול לשמש כיישום מארח עבור מסמכי ActiveX

תוכל לפתוח את מסמך ActiveX גם בדף Web מיוחד שיצרת לשם כך. אם תרצה לראות כיצד ייראה דף HTML לאחר שישולב בו מסמך ActiveX, בחר את הפקודה **Start Browser with URL** והזן את כתובת URL של הדף הרצוי (זכור שכדי שתוכל להשתמש באפשרות זו, יש לשלב בדף Web את הקוד הדרוש להצגת מסמך ActiveX). לחילופין, אם תרצה לנסות את מסמך ActiveX ביישום אחר, תוכל לבחור את הפקודה **Start Program**.

אם הקוד אינו פועל באופן הרצוי, תוכל להיעזר בכל מבחר כלי ניפוי השגיאות של Visual Basic לצורך איתור שגיאות וטיפול בהן (פעולות ניפוי שגיאות נדונו בפרק 9, **יסודות התכנות של Visual Basic**). כך תוכל לשלב נקודות עצירה (Breakpoints), להפעיל את Watch Window שיאפשר לעקוב אחר ערכי משתנים, ולבצע את הקוד שורה אחר שורה. בתרשים 31.7 תוכל לראות פעולה טיפוסית של ניפוי שגיאות.

אזהרה:

סגירת התוכנית ב- Internet Explorer מבלי לסגור את Internet Explorer עלולה לגרום לשגיאה של Internet Explorer. לכן כדאי שתסגור את Internet Explorer ותפעיל אותו מחדש בכל פעם שתרצה להפעיל את המסמך. זכור שסגירת Internet Explorer אינה מפסיקה את פעולת פרויקט המסמך ב- Visual Basic.



תרשים 31.7: שימוש בכלי ניפוי שגיאות יקל עליך באיתור שגיאות ותיקון

הידור המסמך

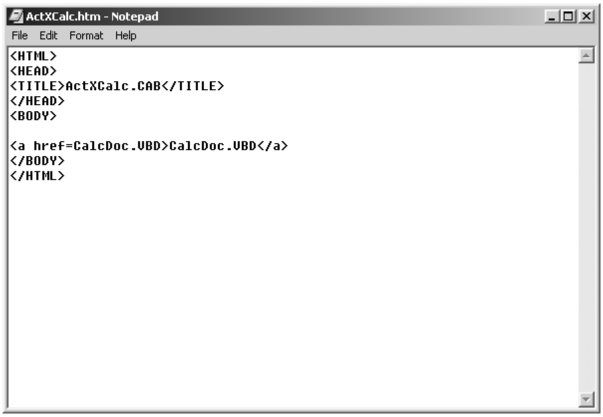
לאחר שתסיים לנסות את המסמך ולנפות ממנו שגיאות, תוכל להדר אותו לצורך הפצה. התחל את התהליך ההידור על ידי בחירה בפקודה **Make** מתפריט **File** של Visual Basic. בתיבת הדו-שיח Make שתוצג תוכל להגדיר את שם ומיקום קובץ EXE או DLL שתיצור. שם קובץ VBD שיווצר, יתבסס על הגדרת המאפיין Name של אובייקט UserDocument. קובץ זה יאוחסן בתיקיה שבחרת לאחסון קובץ EXE.

לאחר שיהודר, המסמך יוכל לשמש בכל תוכנית שתומכת במסמכי ActiveX. קודם לכן הצגנו את Internet Explorer כדוגמה לתוכנית כזו, אוגדן של Office 97 הוא תוכנית נוספת בעלת יכולות אלו. כדי לבצע גישה למסמך ActiveX על ידי שימוש באוגדן, הפעל את האוגדן ובחר את הפקודה **Add From File** מתפריט **Section**. תוצג תיבת דו-שיח שתאפשר לבחור את הקובץ שייטען. הגדר בתיבת הדו-שיח את מיקום קובץ VBD ולחץ על OK כדי לטעון את המסמך.

כדי להעביר את מסמך ActiveX להפצה ב-Web, השתמש ב- Packaging and Deployment Wizard תוך בחירה ב- Internet Package. האשף ייצור תיקיית משנה בשם Package שתכיל את הקבצים הבאים:

- קובץ CAB שיכיל את כל רכיבי ActiveX וקבצי DLL הדרושים לעבודה עם המסמך.
- קובץ VBD שיאפשר לפתוח את המסמך ב- Internet Explorer.
- תיקיית משנה בשם Support במקרה שתצטרך לבנות את קובץ CAB באופן ידני.
- דוגמת קובץ HTML, ראה תרשים 31.8, המציג כיצד לשלב את המסמך בדף Web.

ראה, "אריזת רכיבי ActiveX", נספח 2.



תרשים 31.8: תוכל להעתיק את שורת HREF מקובץ הדוגמה לדף Web שתיצור

חקירת אובייקט UserDocument

כפי שטופס הוא הרכיב המרכזי ביישום רגיל, אובייקט UserDocument הוא הרכיב החשוב ביותר במסמך ActiveX. אובייקט זה הוא ה"בד הציור" שעליו תפרוס את הפקדים שייצרו את ממשק המשתמש של המסמך. ניתן לפרוס פקדים באובייקט UserDocument באותו אופן בו עושים זאת על טופס רגיל, כאשר ניתן להשתמש לחילופין בשיטות גרפיות או בשיטה Print להצגת מידע אחר ישירות על המסמך. יכולת זו מעניקה גמישות רבה בעיצוב המסמכים ובבחירת האופנים להצגת המידע.

הבנת האירועים החשובים של אובייקט UserDocument

אובייקט UserDocument אומנם דומה לטופס במובנים רבים, אך קיימים ביניהם גם מספר הבדלים חשובים. לדוגמה, קיימים מאפיינים, שיטות ומאפיינים שאובייקט UserDocument תומך בהם, והטופס לא, ולהיפך. הבדלים אלה נובעים מההבדל באופי האובייקטים: טופס הוא אובייקט עצמאי, בעוד שאובייקט UserDocument פועל תמיד ביישום מכיל.

האירועים החשובים ביותר שנתמכים על ידי טופס אך אינם נתמכים על ידי אובייקט UserDocument הם: Activate, Deactivate, Load ו-Unload ואילו האירועים שאינם נתמכים על ידי טופס, אך נתמכים על ידי אובייקט UserDocument הם:

- AsycReadComplete. מתרחש כאשר היישום המכיל מסיים ביצוע בקשת קריאה אסינכרונית.
- EnterFocus. מתרחש כאשר מסמך ActiveX מקבל את המוקד.
- ExitFocus. מתרחש כאשר מסמך ActiveX מאבד את המוקד.
- Hide. מתרחש כאשר המשתמש עובר ממסמך ActiveX פעיל למסמך אחר.
- InitProperties. מתרחש כאשר המסמך נטען לראשונה. אך במסמכים שמשתמשים באובייקט PropertyBag לאחסון מאפיינים, מתרחש אירוע ReadProperties גם כאשר המסמך נקרא לראשונה.
- ReadProperties. מתרחש כשפריטים מאוחסנים באובייקט PropertyBag, במקום אירוע InitProperties. האירוע מתרחש גם כאשר מסמך נטען לראשונה.
- Scroll. מתרחש כאשר המשתמש משתמש בפס הגלילה של היישום שמכיל את מסמך ActiveX.
- Show. מתרחש כאשר משתמש מגיע למסמך ActiveX ממסמך אחר.
- WriteProperties. מתרחש בסמוך לסיום פעולת התוכנית. WriteProperties הוא האירוע האחרון המתרחש לפני אירוע Terminate, אך הוא מתרחש רק במקרים שבהם נעשה שימוש בפקודה PropertyChanged, כדי לציין שאירע שינוי בערך המאוחסן באחד המאפיינים.

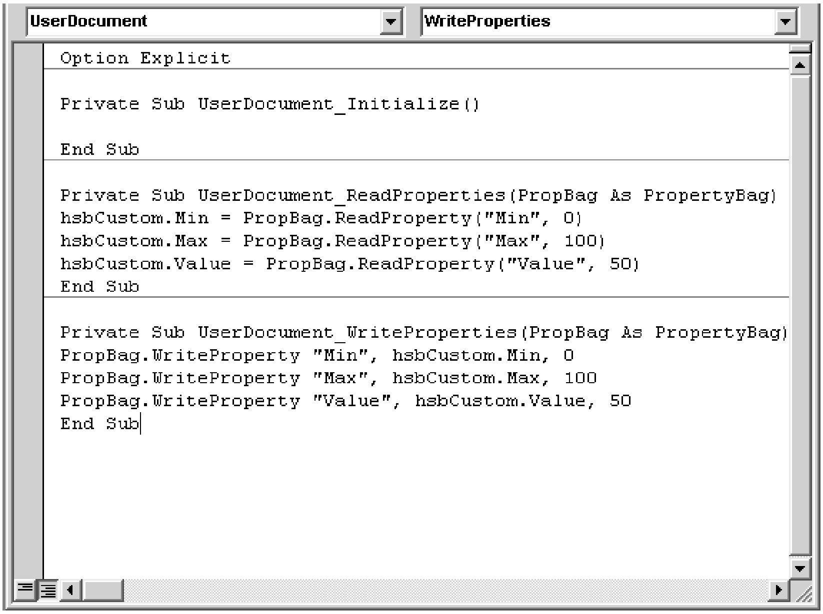
יצירה ואחסון מאפיינים של אובייקט UserDocument

בין אובייקט UserDocument לבין טופס יש נקודות דמיון רבות, אך יש גם תחומים שבהם אובייקט UserDocument דומה יותר לאובייקט UserControl מאשר לטופס. כל שלושת סוגי האובייקטים, UserDocument, טופס ו-UserControl מאפשרים ליצור מאפיינים ושיטות כדי להרחיב את מיגוון היכולות שלהם. אך רק לאובייקטים מסוג UserDocument ו-UserControl יש יכולת לשימוש באובייקט PropertyBag. אובייקט PropertyBag, בשילוב עם כמה אירועים המיוחדים לו, משמש לשמירה על ערכי מאפיינים ציבוריים בזמן שבין הפעלות האובייקטים.

משום שהתהליך המשמש ליצירת מאפיינים ולשמירה על הערכים שהוצבו בהם תואר כבר בפירוט (ראה פרק 14), לא נחזור על כל פרטי התהליך. אך נביא כעת סקירה קצרה של המרכיבים העיקריים שלו, כדי לסייע לך להיזכר בפעולות. כדי ליצור מאפיינים לאובייקט UserDocument ולשמור את ערכיהם, בצע את הפעולות הבאות:

1. צור מאפיין על ידי שימוש בפרוצדורות Property Let ו- Property Get. את מעטפת המאפיין תוכל ליצור בעזרת תיבת הדו-שיח Add Procedure, שאליה תוכל לגשת באמצעות תפריט **Tools** של Visual Basic.
2. כדי לציין שערך המאפיין שונה, הצב פקודת PropertyChanged בפרוצדורת Property Let של כל מאפיין שאת ערכו תרצה לאחסן.
3. לאחסון ערכי מאפיינים, היעזר בשיטה WriteProperty של אובייקט PropertyBag לכתיבת הערכים החדשים של המאפיינים שערכיהם שונו. הצב את קוד שיטה זו באירוע WriteProperties של אובייקט UserDocument.
4. לאחזור ערך המאפיין, תוכל להיעזר בשיטה ReadProperty של אובייקט PropertyBag כדי לשמור את ערכי המאפיינים שערכיהם שונו. הצב את קוד שיטה זו באירוע ReadProperties של אובייקט UserDocument.

ראה, "הוספת מאפיינים", פרק 14.



תרשים 31.9: היעזר בשיטות ReadProperty ו-WriteProperty לשמירה ואחזור ערכי מאפיינים ציבוריים של אובייקט UserDocument

שיטות אובייקט UserDocument

אובייקט UserDocument תומך גם בשתי שיטות חשובות שאינן נתמכות על ידי טופס: AsyncRead ו-CancelAsyncRead. השיטה AsyncRead מאפשרת למסמך להעביר בקשה שהיישום שמכיל אותו יקרא נתונים מקובץ או מכתובת URL. משם השיטה משתמע שפעולת הקריאה תתבצע באופן אסינכרוני. קל להבין שיכולת לקריאה אסינכרונית של נתונים מועילה מאוד בעבודה ב-Web. לדוגמה, אתה עשוי לכתוב יישום להצגת נתונים אשר יקבל את הנתונים משרת. השיטה AsyncRead תדרוש ממך להגדיר קובץ (או מקור מידע) שייקרא ואת סוג הנתונים שייקראו. שלושת סוגי הנתונים שבהם תומכת השיטה יפורטו בטבלה 31.2.

טבלה 31.2: סוגי נתונים הנתמכים על ידי השיטה AsyncRead

קבוע	תיאור
vbAsyncTypeFile	הנתונים בקובץ שנוצר באמצעות Visual Basic
vbAsyncTypeByteArray	הנתונים שייקראו לקוחים מתוך Byte Array שמכיל את הנתונים שיאוחזרו
vbAsyncTypePicture	הנתונים מאוחסנים באובייקט Picture (תמונה)

השיטה CancelAsyncRead משמשת להפסקת הקריאה האסינכרונית לפני השלמתה.

לצורך הדגמה, נוסיף כעת קריאה אסינכרונית למסמך ActiveX, Loan Calculator. לפני שתוסיף את קוד השיטה AsyncRead בצע את פעולות ההכנה הבאות:

1. צור את פרויקט הדוגמה שתואר בסעיפים הקודמים.
2. פתח את אובייקט UserDocument, CalcDoc והוסף לטופס שלו שני פקדים נוספים, לחצן פקודה ופקד Image.
3. קרא ללחצן הפקודה: cmdGetPic, ולפקד התמונה: imgMain.
4. מצא את הנתיב לקובץ תמונה המאוחסן בדיסק או ב-Web שתשתמש בו לצורך הניסוי. לצורך הדוגמה השתמשתי בנתיב D:\Pictures\Brian.BMP.
5. הוסף את קוד השיטה AsyncRead לשגרת האירוע Click של לחצן הפקודה:

```
Private Sub CmdGetPic_Click( )
 AsyncRead "d:\pictures\brian.bmp", vbAsyncTypePicture, "MyPicture"
End Sub
```

קטע קוד זה יחל הורדת קובץ התמונה BRIAN.BMP למחשב הלקוח. בדוגמה כללתי הגדרת נתיב מפורט לקובץ המאוחסן בדיסק הקשיח במחשב מקומי, אך גם יכולתי להשתמש בכתובת URL כגון http://myserver/mypicture.gif.

6. הוסף את קטע הקוד הבא לשגרת האירוע AsyncReadComplete של אובייקט UserDocument, כדי להציג את התמונה שהורדת זה עתה בפקד Image :

```
Private Sub UserDocument_AsyncReadComplete(AsyncProp As AsyncProperty)

MsgBox "The picture downloaded has finished with status " & AsncProp.StatusCode
Set imgMain.Picture = AsyncProp.Value
End Sub
```

אירוע AsyncReadComplete מאותחל לאחר סיום הורדת התמונה וגורם להצגת תיבת הודעה, והתמונה (שמאוחסנת בפרמטר AsyncProp) מוצגת בפקד Image.

הערה:

בטעינת תמונה מה-Web (בשימוש בתחביר //:http), Internet Explorer עשוי לאחסן את התמונה במטמון המחשב המקומי. אם תרצה לנצל את האחסון באופן מיטבי, הוסף פרמטר אופציונלי לסוף שורת הקריאה לשיטה AsyncRead, שיגדיר האם ובאילו נסיבות ייעשה שימוש בתמונה המאוחסנת. את רשימת הערכים שתוכל להציב בפרמטר האופציונלי תוכל למצוא בנושא AsyncRead method בעזרה המקוונת.

חשוב לזכור שבדוגמה תתרחש רק פעולת הורדה אסינכרונית אחת. אם תיזום מספר פעולות הורדה שתתבצענה במקביל, תוכל להיעזר במאפיין PropertyName כדי לקבוע איזה אובייקט יועבר לאירוע AsyncReadComplete. במקרה המוצג בדוגמה, הערך המאוחסן במאפיין זה הוא myPicture.

בנוסף ליזימת פעולות קריאה אסינכרוניות ולזיהוי מצבי הסיום שלהן, Visual Basic גם מאפשרת לנהל מעקב אחר הפעולות באמצעות האירוע AsyncReadProgress. לאירוע מועברים כמה מאפיינים המאפשרים להציג את מידת התקדמות הפעולה. לדוגמה, ייתכן שתרצה להציג פקד מד התקדמות שיציג את מספר הבתים שהורדו עד כה, או שתרצה להציג הודעות שתיידענה על יצירת החיבור ותחילת הורדת הקובץ.

אובייקט Hyperlink במסמכים

אובייקט Hyperlink הוא אובייקט חשוב במסמכי ActiveX. לאובייקט זה ניתן לגשת על ידי שימוש במאפיין HyperLink של אובייקט UserDocument. האובייקט אינו כולל מאפיינים והוא מכיל רק שלוש שיטות, אך הוא מאפשר למסמכי ActiveX לקרוא למסמכי ActiveX אחרים, או לבצע פעולות ניווט באתרי Web. שלוש השיטות הן :

- NavigateTo. השיטה גורמת ליישום המכיל לעבור לקובץ או כתובת URL המצוינים בשיטה. ניתן להשתמש בשיטה זו למעבר ממסמך ActiveX אחד לאחר.
- GoBack. השיטה עוברת למסמך קודם ברשימת History של היישום המכיל. אם היישום אינו תומך ב-Hyperlink, או שרשימת History ריקה, תתרחש שגיאה.
- GoForward. שיטה זו מנוגדת לשיטה GoBack. השיטה גורמת ליישום המכיל לעבור למסמך הבא ברשימת History. אם לא קיים כזה, תתרחש שגיאה.

הערה:

יישום מכיל שתומך בפעולות Hyperlink, כדוגמת Internet Explorer הוא שמבצע את הפעולה הדרושה למעבר למסמך שצוין בשיטה NavigateTo. יישום שאינו תומך בפעולות Hyperlink, כדוגמת האוגדן של Office 97 מפעיל יישום שתומך בפעולות כאלו, כדי שיבצע עבורו את הפעולה.

בשורות הקוד המובאות להלן דוגמאות לשימוש בשיטה NavigateTo :

```
UserDocument.Hyperlink.NavigateTo "MyDoc.VBD"
UserDocument.Hyperlink.NavigateTo "http://www.mysite.com"
```

השורה הראשונה תטען מסמך ActiveX אחר המאוחסן באותה תיקיה, בעוד שהשנייה תפנה את היישום המכיל לאתר ב-Web.

ActiveX Document Migration Wizard

עד כה למדת רק כיצד ליצור מסמכי ActiveX מאפס. אך ייתכן מאוד שכבר הקדשת זמן רב ומאמצים רבים ליצירת יישומים רגילים ועל כן אתה תוהה האם תוכל לבצע פעולות גזירה והדבקה ולנצל חלקים מתוכניות קיימות?

למרבה המזל התשובה חיובית. Visual Basic מציעה כלי שנקרא ActiveX Document Migration Wizard שיסייע בהסבת טפסים הקיימים ביישומים לאובייקטי UserDocument שאותם תוכל לשלב במסמכי ActiveX. מילת המפתח במשפט האחרון היא "יסייע". האשף לא ייצור מסמך ActiveX שלם מהיישום הקיים שלך. במקום זאת, הוא יבצע את הפעולות הבאות :

- יעתיק מאפייני טפסים לאובייקטי UserDocument חדשים.
- יעתיק פריטי תפריטים מטפסים מקוריים לאובייקטי UserDocument חדשים.
- יעתיק פקדים מטפסים קיימים וימקם אותם באותו מיקום יחסי על המסמכים, זאת תוך שמירה על כל הגדרות מאפייני הפקדים. פקדי OLE Container ואובייקטי OLE שהוטבעו ביישומים לא יועתקו.
- יעתיק קוד משגרות האירוע של הטופס לשגרות התואמות של מסמך ActiveX, כולל הקוד שבשגרות האירוע של הפקדים.
- פקודות שאינן נתמכות על ידי מסמכי ActiveX כגון Load, Unload ו-End תהפוכנה להערות.

ActiveX Document Migration Wizard יכול אומנם לבצע את רוב הפעולות הדרושות להסבת היישום למסמך ActiveX, אך קיימות גם פעולות רלוונטיות שהוא אינו יכול לבצע עבורך. על כן יהיה עליך לבצע פעולות כתיבת קוד, בטרם תוכל להדר את המסמך ולהפיץ אותו.

ראשית תצטרך לגרוע מהיישום הקיים אירועים שאינם נתמכים על ידי פקדי ActiveX כגון Load ו-Unload. האשף אומנם הופך פקודות Load ו-Unload להערות, אך הוא אינו משנה את הקוד בשגרות האירוע של האירועים הרלוונטיים. אם כללת בשגרות אירוע אלו קוד שמאתחל את מאפייני הטופס או הפקדים, תרצה אולי להעביר חלק מהקוד שבשגרת אירוע Load לשגרת האירוע Initialize של אובייקט UserDocument. ייתכן גם שתרצה להעביר קוד שבשגרת האירוע Unload לשגרת האירוע Terminate של אובייקט UserDocument.

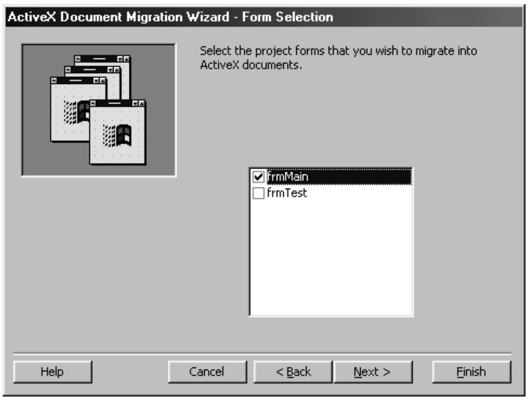
תצטרך גם לוודא שהקוד אינו כולל פניות לאובייקטים שאינם קיימים יותר. לדוגמה, לא ניתן לשלב באובייקט UserDocument שנוצר על ידי הסבת טופס, אזכורים כלשהם לשם אותו טופס.

הרצת ActiveX Document Migration Wizard

כדי שתוכל להפעיל את ActiveX Document Migration Wizard, תצטרך לוודא שהוא זמין לשימוש מ- Visual Basic. כדי להפוך את האשף לזמין בחר בו בתיבת הדו-שיח Add-In Manager, שאותה תציג על ידי בחירה ב- **Add-In Manager** מתפריט **Add-Ins**. לאחר שתוסיף את האשף לסביבת הפיתוח של Visual Basic, תוכל להפעיל אותו על ידי בחירת **ActiveX Document Migration Wizard** מתפריט **Add-In**.

הצעדים הבאים יפרטו את התהליך להסבת טפסים למסמכי ActiveX:

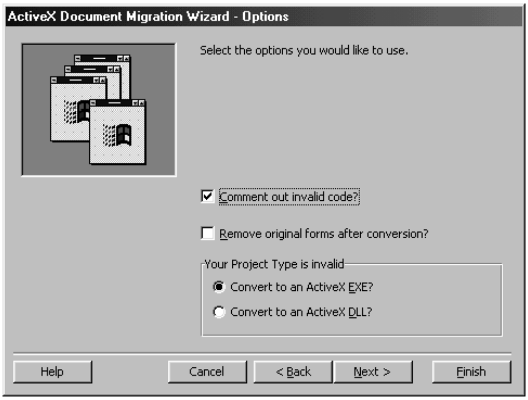
1. פתח את הפרויקט שאת הטפסים שבו ברצונך להסב. האשף יפעל בצורה תקינה רק מתוך הפרויקט שעליו הוא פועל.
2. הפעל את האשף. תחילה יוצג מסך הקדמה, שיציג פירוט קצר של פעולות שהאשף יבצע ולא יבצע. לחץ על לחצן Next כדי להמשיך.
3. מהמסך המוצג בתרשים 31.10 בחר את הטפסים שאותם ברצונך להסב. ברשימת הטפסים יפורטו כל הטפסים בפרויקט הפעיל. תוכל לבחור טופס על ידי סימון התיבה שלצד שמו. ניתן לבחור מספר טפסים בבת-אחת. לאחר שתסיים לבחור את הטפסים, לחץ על לחצן Next כדי לעבור לשלב הבא.



תרשים 31.10: בחר טפסים מתוך הרשימה שתוצג באשף

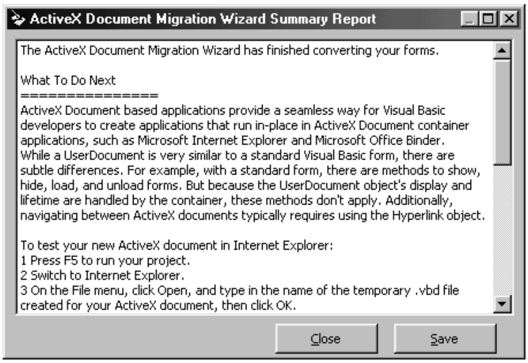
4. היעזר במסך Options של האשף (ראה תרשים 31.11) כדי להגדיר את אופן עיבוד הטפסים. שלוש האפשרויות המוצעות תאפשרנה לבצע את הפעולות הבאות:

- להפוך קטעי קוד בלתי חוקיים להערות. בחירה באפשרות זו תהפוך פקודות שאינן נתמכות על ידי מסמכי ActiveX, כגון Load, Unload ו-End להערות.
- לגרוע את הטפסים מהפרויקט לאחר שיוסבו. בחירה באפשרות זו תמחק טפסים מהפרויקט הפעיל לאחר פעולות ההסבה. ברוב המקרים לא תבחר אפשרות זו, מפני שתרצה להותיר את הפרויקט המקורי שלם.
- האם הפרויקט יוסב לקובץ ActiveX EXE או לקובץ ActiveX DLL. ערך ברירת המחדל של אפשרות זו הוא קובץ ActiveX EXE (קבצי ActiveX DLL משמשים ליצירת רכיבים משותפים ולא ליצירת יישומים).



תרשים 31.11: בחר את האפשרויות העונות על צרכיך

לאחר שתבחר את האפשרויות, לחץ על לחצן Next כדי לעבור לדף האחרון.

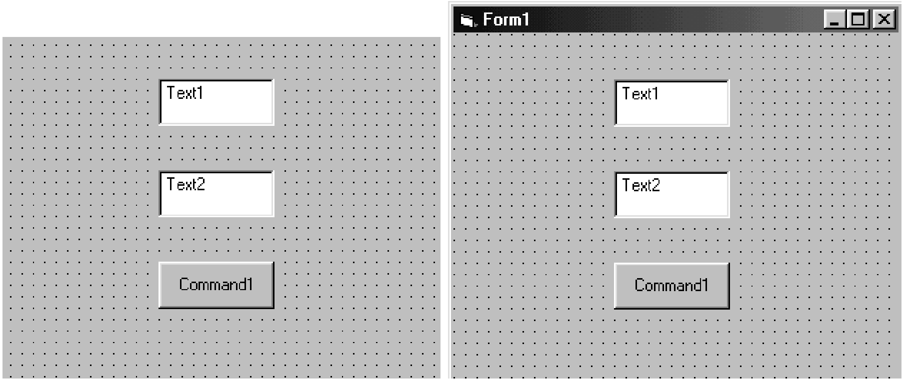


תרשים 31.12: ActiveX Document Migration Wizard מפיק דוח סיכום אשר בו תפורטנה הפעולות שתידרש לבצע כדי להשלים את תהליך ההסבה

5. בחר האם תרצה להציג דוח סיכום פעולה, לאחר השלמת תהליך ההסבה שיבוצע על ידי האשף. לאחר שתבחר את האפשרות הרצויה לך, לחץ על לחצן Finish, כדי להתחיל את פעולת ההסבה. דוח הסיכום המוצג בתרשים 31.12 יפרט את הפעולות הנוספות שתידרש לבצע כדי להשלים את תהליך ההסבה.

הצגת התוצאות

לאחר ש- ActiveX Document Migration Wizard יסיים את פעולתו הוא יאחסן את אובייקטי UserDocument החדשים שנוצרו על-ידו בפרויקט שבו נכללו הטפסים המקוריים. קבצי קוד המקור של המסמך יאוחסנו בתיקיה בה אוחסנו טפסי הפרויקט המקורי, תוך שימוש בשמות הטפסים המתאימים עם סיומות מתאימות. לדוגמה, הסבת טופס שיאוחסן בקובץ FRMTEST1.FRM תגרום ליצירת אובייקט UserDocument שיאוחסן בקובץ DOCTEST1.DOB. כבר ציינו שפקדים שנכללו בטפסים יועתקו לאובייקטי UserDocument, תוך שמירה על מיקום יחסי. בתרשים 31.13 תוכל לראות את הטופס המקורי ואת אובייקט UserDocument שיווצר ממנו.



תרשים 31.13: ממשק המשתמש של אובייקט UserDocument זהה לזה של הטופס המקורי שממנו נוצר

כפי שציינו קודם לכן, רוב הקוד שהיה בטפסים המקוריים יועתק לאובייקט UserDocument. קוד שאינו חוקי לשימוש במסמכי ActiveX יזוהה על ידי האשף ויוסב באופן אוטומטי להערות (פעולה זו תבוצע רק אם בחרת את האפשרות Comment Out Invalid Code במסך Options של האשף). בתרשים 31.14 תוכל לראות קטע קוד שבוצעה עליו פעולת הסבת קוד בלתי חוקי להערות.



תרשים 31.14: ניתן לזהות קוד בלתי חוקי שהפך להערה על-פי מילת המפתח [AXDW]

יצירת מסמך מורכב

השטח הפנוי באובייקט UserDocument מוגבל ולכן תוגבל בכמות המידע שתוכל להציג במסמך יחיד. בהקשר זה מסמך יחיד מקביל ליישום המכיל טופס יחיד. אך תוכל גם ליצור מסמכים נוספים שייכללו בפרויקט, ולנווט ביניהם. תוכל גם לשלב טפסים רגילים ביישומים שיכילו מסמכי ActiveX. בסעיפים הבאים נסקור את הפרטים הקשורים לשימוש במסמכי ActiveX ובטפסים מרובים ביישום יחיד, המבוסס על מסמכי ActiveX.

פיתוח מסמכים נוספים

כדי שתוכל להשתמש במסמכים נוספים ביישום המבוסס על מסמך ActiveX, יהיה עליך ליצור מסמכים נוספים ולהגדיר מנגנון שיאפשר לעבור ביניהם. התהליך שונה מעט מזה המשמש למעבר בין טפסים, מפני שאובייקט UserDocument אינו תומך בפקודות Load ו-Unload ובשיטות Show ו-Hide שזמינות בעבודה עם טפסים.

כדי להוסיף מסמך נוסף לפרויקט, הוסף לפרויקט אובייקט UserDocument נוסף, על ידי בחירה בפקודה **Add User Document** מתפריט **Project**. הפעולה תוסיף אובייקט UserDocument נוסף לפרויקט, ותאחסן אותו בתיקיה UserDocuments. יש להגדיר שם שתחתיו יאוחסן האובייקט החדש, באותו אופן בו עשית זאת לאובייקט UserDocument הראשון שצורף לפרויקט.

אחר תידרש לשרטט את ממשק המשתמש של אובייקט UserDocument החדש, ולהוסיף לפרויקט את הקוד שיאפשר לאובייקט זה לפעול. הוסף גורמים אלה על ידי ביצוע אותן פעולות שביצעת באובייקט UserDocument הראשון.

כעת הגענו ל"מלכודת". היות ומסמכי ActiveX אינם תומכים בשיטה Show (שנתמכת על ידי טפסים), נשאלת השאלה כיצד תעבור בין המסמכים השונים שייכללו ביישום? התשובה היא על ידי שימוש בשיטה NavigateTo של אובייקט Hyperlink. השיטה מורה ליישום המכיל לעבור לקובץ או לכתובת URL ולטעון מתוכם קובץ. אם הקובץ שייטען יהיה דף Web, הוא יעובד באופן דומה לאופן בו עובד הדף המקורי.

למעבר ממסמך ראשון למסמך שני, תוכל להשתמש בשורת הקוד המוצגת להלן:

```
Hyperlink.NavigateTo App.Path & "\docnav2.vbd"
```

המאפיין App.Path מכיל את הגדרת נתיב הגישה למסמך הפעיל. השתמש בנתיב זה כנקודת מוצא לאיתור המסמך השני ותוכל לטעון אותו ללא בעיות, זאת בהנחה שהמסמכים מאוחסנים באותה תיקיה. כדי לחזור מהדף השני לראשון, השתמש שוב בשיטה NavigateTo. השיטה משמשת בקטעי קוד לתגובה לאירועים בשני המסמכים. ברוב המקרים האירוע שבקוד שלו שולבה קריאה לשיטה הוא אירוע Click.

כאשר תעבור למסמך השני, שים לב שלחצן Back של Internet Explorer יופעל. תוכל להשתמש בלחצן זה כדי לשוב למסמך הראשון. אך למרות זאת כדאי שתשלב במסמך השני קישור ישיר למסמך הראשון, שייבנה תוך שימוש בשיטה NavigateTo, מפני

שייתכן שהמסמך הראשון לא יהיה הפריט הקודם ברשימת ההיסטוריה של היישום המכיל. במצבים שבהם המסמך הראשון כלל אינו נמצא ברשימת ההיסטוריה, השיטה NavigateTo עשויה להיות האמצעי היחיד שיאפשר לחזור אליו.

שימוש והצגת טפסים מהמסמך

בנוסף לעבודה עם מספר מסמכים במקביל, תוכל לעבוד עם טפסים רגילים ביישומים שיתמכו במסמכי ActiveX. אם תרצה להשתמש בטופס בפרויקט של מסמך ActiveX, צור אותו באותו אופן בו יצרת טפסים בפרויקטים רגילים - על ידי שרטוט ממשק משתמש על הטופס וכתיבת קוד שיבצע את הפעולות הרצויות. כדי להציג את הטופס מתוך המסמך, תוכל להשתמש בשיטה Show. לאחר מכן, כאשר תרצה להסתיר את הטופס, תוכל להיעזר בפקודה Unload.

ראה, "אבני היסוד של Visual Basic", פרק 3.

טפסים אומנם יכולים להיות חלק מיישומים, אך הם אינם מטופלים באותו אופן בו מטופלים מסמכים. טפסים אינם מוכלים ביישום שמכיל את מסמך ActiveX, ואינם תלויים ביישום המכיל.

מכאן...

פרק זה הציג מסמכי ActiveX. למדת כיצד מסמכים כאלה מאפשרים ליצור יישומים שיפעלו ב- Internet Explorer או ביישום מכיל אחר שתומך ב-ActiveX. למדת גם על הדמיון הרב שבין אובייקט UserDocument לבין טופס ביישום רגיל.

במהלך הפרק גם נגענו במספר נושאים הקשורים ליצירת מסמכי ActiveX שנדונו ביתר פירוט בפרקים אחרים. מידע נוסף על נושאים אלה תוכל למצוא בפרקים הבאים:

- מידע נוסף אודות יצירה ועיצוב טפסים תוכל למצוא בפרק 3 **אבני היסוד של Visual Basic**.
- מידע נוסף אודות פעולות לניפוי שגיאות מתוכניות תוכל למצוא בפרק 9 **יסודות התכנות של Visual Basic**.
- מידע נוסף אודות יצירת פקדי ActiveX תוכל למצוא בפרק 14 **יצירת פקדי ActiveX**, ובפרק 15 **הרחבת פקדי ActiveX**.
- מידע אודות שיטה אלטרנטיבית ליצירת יישומים המיועדים לפעול בסביבת ה-Web תוכל למצוא בפרק 31 **דפי שרת פעילים**.

32

Visual Basic ושימושים נוספים באינטרנט

מה בפרק?

- **הוספת יכולות דפדפן ליישומים**
- **תכנות דואר אלקטרוני**
- **פקד Internet Transfer**

מרבית הפרקים שנכללו בחלק זה של הספר שעוסק ב-Web, התמקדו בשימוש ב- Visual Basic כדי להוסיף לדפי Web יכולות החורגות מעבר לאלו שניתן לממש בדפי HTML רגילים. דפי Web הם אולי אופן השימוש הפופולרי ביותר ב-Web, אך השימוש ב- Visual Basic בסביבת Web מאפשר לעשות דברים נוספים רבים, כגון הפעולות המפורטות להלן:

- שילוב דפדפן Web ביישומים.
- משלוח וקבלת הודעות דואר אלקטרוני.
- העברת קבצים על ידי שימוש בפקד Internet Transfer.

רשימה זו אינה רשימה מלאה של האופנים שבהם ניתן להשתמש ב- Visual Basic ב-Web, אך הנושאים שנכללו בה עשויים להיות רלוונטיים למפתחים רבים. פרק זה יהווה מבוא מעשי לנושאים שברשימה.

הוספת יכולות דפדפן ליישומים

במקרים רבים תרצה לשלב יכולות דפדפן Web בתוכניות רגילות ב- Visual Basic. לדוגמה, ייתכן שתרצה לשלב קישורים לדפי Web שיכילו תיעוד של היישום, או מידע מועיל אחר. תוכל גם לשלב ביישומים פקדי Browser שיאפשרו למשתמשים להציג דפי Web בחלון המוכל בטופס של Visual Basic. בעזרת שני אופני פעולה אלה תוכל לשלב יכולות דפדפן ביישומי Visual Basic.

יצירת דפדפן בטופס

הדרך הפשוטה ביותר בה תוכל להוסיף תוכן המאוחסן באתר Web ליישום, היא על ידי שימוש בפקד WebBrowser שמותקן יחד עם Internet Explorer 4.0. תחילה יהיה עליך להוסיף את הפקד לפרויקט רגיל מסוג EXE, על ידי בחירה בפקודה **Components** מתפריט **Project** ובחירה בפריט **Microsoft Internet Controls** מתיבת הדו-שיח שתוצג לפניך. לאחר מכן יהיה עליך לשרטט פקד WebBrowser על הטופס ולהגדיר את הגודל הרצוי.

כדי לאפשר לפקד WebBrowser לפעול, כתוב שורת קוד יחידה:

```
WebBrowser1.Navigate "http://www.quecorp.com"
```

השיטה Navigate של פקד WebBrowser גורמת לפקד להתחבר לכתובת URL שפורטה בפקודה ולאחזר ממנה דף Web כמתואר בתרשים 32.1.

משום שפקד WebBrowser פועל בתוכנית Visual Basic, הוא אינו מכיל כברירת מחדל את סרגל הכלים והפקדים האחרים שקשורים ב- Internet Explorer. אך תוכל להוסיף לפקד לחצני פקודה משלך ופונקציות מותאמות אישית, שתאפשרנה לו לנווט לאתרים הרצויים.



תרשים 32.1: פקד WebBrowser מאפשר לשלב דפי Web "חיים" בטפסים

דפי Web עשויים להכיל גם אלמנטים גרפיים ופריטים אחרים, שהורדתם מה-Web עשויה להימשך זמן רב. הפעולות תבוצענה על ידי פקד WebBrowser באופן אסינכרוני מבלי להפריע לתוכנית להמשיך בפעולה רגילה. אך סביר להניח שתרצה לדעת מתי תסתיים הורדת דף מסוים. בוודאי שמת לב שהטופס המוצג בתרשים 32.1 מכיל תיבת רשימה שבה מוצג דיווח על מצב פעולת הדפדפן. תיבת הרשימה מנוהלת על ידי קטע קוד שפונה למספר מאפיינים ואירועים של פקד WebBrowser:

- המאפיין Busy של הפקד הוא מאפיין בוליאני שמכיל ערך True בכל עת שהפקד מצוי במצב תקשורת עם שרת Web. ביישום הדוגמה המוצג בתרשים 32.1 נעשה שימוש בפקד Timer לצורך בדיקת ערך המאפיין ועדכון התצוגה בתיבת הרשימה.
- אופן הפעולה של שיטת Stop דומה לזה של לחצן Stop שבסרגל הכלים של Internet Explorer. השיטה יכולה לשמש לעצירת פעולת הפקד, כשהורדת הדף נמשכת זמן רב מדי.
- אירוע DocumentComplete הוא אירוע שמאותחל על ידי פקד WebBrowser לאחר שהאתר הרצוי מוצג בשלמותו. האירוע לא יאותחל אם מאפיין Visible של הפקד יהיה False.

הקוד שנדרש ליצירת הדוגמה שבתרשים 32.1 יובא בתוכנית 32.1.

תוכנית 32.1: BROWSER.VBP - שימוש בפקד WebBrowser.

```
Private Sub cmdNavigate_Click( )
    WebBrowser1.Visible = False
    WebBrowser1.Navigate txtURL.Text
    tmrMain.Intrval = 2000
    tmrMain.Enabled = True
End Sub
```

```
Private Sub tmrMain_Timer( )
  If WebBrowser1.Busy = True Then
      lstStatus.AddItem "Working..."
  Else
      WebBrowser1.Visible = True
      tmrMain.Enabled = False
  End If
End Sub

Private Sub WebBrowser1_DocumentComplete(ByVal pDisp As Object, URL As Variant)
   lstStatus.AddItem "Document display completed. "
End Sub
```

הפעלת הדפדפן מהיישום

בנוסף לפקד WebBrowser, תוכל גם להפעיל את הדפדפן כתוכנית נפרדת. אופן הפעולה יאפשר למשתמש יותר חופש פעולה מפני שהוא יאפשר לו לשנות את גודל חלון הדפדפן ולהשתמש בכלים המוצעים בדפדפן רגיל. יתרון נוסף הוא ששיטה זו אינה צורכת שטח יקר בטופס, כפי שקורה עם פקד WebBrowser שמוטבע בטופס.

יצירת קיצור דרך לאינטרנט

הצעד הראשון למעבר לאתרי Web מסוימים הוא הבנת המושג **קובץ URL**, המכונה גם **קיצור דרך לאינטרנט** (Internet Shortcut). קיצורי דרך לאינטרנט דומים לקיצורי דרך רגילים, אך הם מצביעים על אתרי Web במקום על קבצי יישומים. לחיצה על קיצור דרך לאינטרנט פותחת את הדפדפן ומנווטת לאתר שקיצור הדרך מייצג. קיצור דרך לאינטרנט הוא קובץ טקסט פשוט (למעשה קובץ INI), שמכיל את כתובת URL של האתר הרצוי. המבנה הקובץ הוא כדלקמן:

```
[InternetShortcut]
URL=http://www.que.com/
```

רשימת Favorites של Internet Explorer היא למעשה אוסף קבצי URL. אתר את התיקיה Favorites בדיסק הקשיח ותוכל לראות את קבצי URL. המבנה הפשוט של קבצי URL מקל על יצירתם ב- Visual Basic. תוכל ליצור אותם באמצעות פונקציות לעבודה עם קבצי INI שתוארו בפרק 21, או על ידי יצירת קובץ טקסט פשוט:

```
Open "C:\TEMP.URL" For Output As #1
Print #1, "[InternetShortcut]"
Print #1, "URL=http://www.que.com/"
Close #1
```

קטע הקוד שהובא, ייצור קיצור דרך לאינטרנט בשם TEMP.URL. לאחר שתיצור אות, תוכל ללחוץ על הסמל שלו כדי להיכנס לאתר Web שעליו הוא מצביע.

ראה: "שימוש בקבצי INI", פרק 21.

הפעלת קיצור דרך ב-Web

יש כמה דרכים שמאפשרות להפעיל קיצור דרך לאינטרנט מתוך Visual Basic. תוכל לבצע קריאה ל-API, או לעבור לתוכנית אחרת, שתפתח עבורך את הדפדפן.

קריאה לפונקציית API, ShellExecute תפעיל את התוכנית שקושרה לסיומת URL. אם כל התוכניות הרלוונטיות הותקנו כנדרש במערכת שלך, תוכנית זו תהיה דפדפן ברירת המחדל שלך. בדומה לפונקציות API אחרות, תצטרך להצהיר על הפונקציה ShellExecute בטרם תוכל לקרוא לה.

ראה: "שימוש בפונקציות API ב- Visual Basic", פרק 20.

קטע הקוד הבא יעלה אתר Web, תוך שימוש בפונקציה ShellExecute ובקובץ C:\TEMP.URL :

```
Const SW_SHOWNORMAL = 1
Dim lRetVal As Long ' Return Value
Dim lWindow As Long 'BROWSER HWND
lRetVal = ShellExecute(lWindow, "open", "C:\Temp.URL", "", "", SW_SHOWNORMAL)
```

הערה:

פונקציות API, ShellExecute ו-FindExecutable יכולות לשמש לבדיקה מי התוכנית שקושרה לסיומת מסוימת ולהפעלתה. פונקציות אינן מוגבלות רק לשימוש בקיצורי דרך לאינטרנט, וזוהי הסיבה שבעטיה פרמטרים כה רבים נותרו ריקים בדוגמה.

השימוש בפונקציית API, ShellExecute לשם כניסה לאתר Web מסוים הוא אומנם פשוט מאוד, אך קיימת דרך עוד יותר פשוטה לעשות זאת.

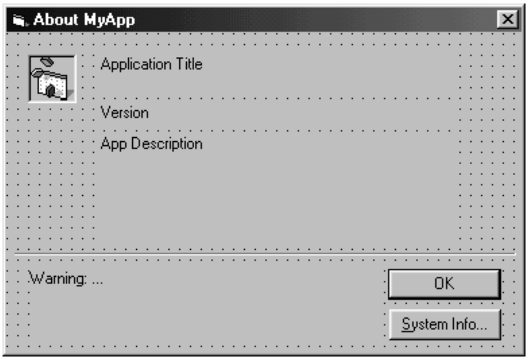
תיבת About משופרת

כדי להמחיש כיצד ניתן לפתוח אתר Web מסוים מתוך Visual Basic, ניצור תיבת About שתכיל קישור לאתר Web. תיבת About היא טופס (שברוב המקרים ניתן להציג אותו על-ידי בחירה בפקודת התפריט About) בו מוצג מידע אודות התוכנית. Visual Basic מציעה תבנית לתיבת About שעליה נבסס את הפרויקט שלנו.

בצע את הפעולות הבאות כדי ליצור את יישום הדוגמה שיציג תיבת About :

1. צור פרויקט חדש מסוג Standard EXE.
2. לחץ על לחצן **Add Form** או בחר את הפקודה **Add Form** מתפריט **Project** כדי להציג את תיבת הדו-שיח **Add Form**.

3. בחר מתיבת הדו-שיח את הפריט **About Dialog** ולחץ על לחצן **Open**. לתוכנית יתוסף טופס חדש שייראה כמו הטופס המוצג בתרשים 32.2.



תרשים 32.2: Visual Basic מציעה טופס סטנדרטי לתיבת About

4. הקטן את תווית תיאור היישום lblDescription כדי לפנות מקום לפקד תווית נוסף על הטופס (למשל, שנה את הגדרת מאפיין Height של התווית ל-550).

5. שרטט פקד תווית נוסף מתחת לפקד lblDescription וקרא לו lblURL.

6. הצג את חלון המאפיינים של פקד lblURL ושנה את הגדרות המאפיינים כלהלן:

   - הגדר את המאפיין ForeColor כ-Light Blue.
   - הצג את תיבת הדו-שיח Font והפעל את המאפיין Underline.
   - הצב במאפיין Caption של הפקד את הכתובת: www.quecorp.com.

   לאחר שתבצע את הפעולות, התווית תראה כמו Hyperlink.

7. הוסף את הקוד המובא בתוכנית 32.2 לשגרת אירוע Click של פקד התווית. הקוד משמש לפתיחת כתובת URL.

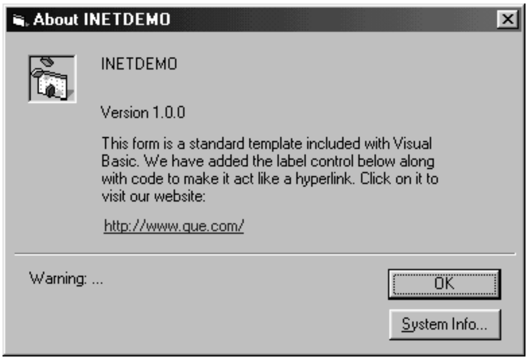
תוכנית 32.2: INETDEMO.VBP - פתיחת אתר Web מתוך טופס.

```
Private Sub lblURL_Click( )
       Shell "C:\windows\explorer.exe  http://www.hod-ami.co.il", vbMaximized
End Sub
```

כדי לנסות את התוכנית, הצג את תיבת About ולחץ על כתובת URL שתוצג בה. בדרך כלל תיבת About מוצגת בעקבות בחירה בפקודה About מהתפריט. אך לצורך ניסוי התוכנית, תוכל לשלב בשיגרה Form_Load את שורת הקוד הבאה:

```
frmAbout.Show
```

תיבת About המשופרת מוצגת בתרשים 32.3.



תרשים 32.3: פקד Label שעוצב נכון עשוי לשמש כ-Hyperlink "מזויף"

תכנות דואר אלקטרוני

כיום מתקבל הרושם שלכל אדם כמעט יש **דואר אלקטרוני** (E-Mail). דואר אלקטרוני הוא אמצעי אידיאלי להעברת מסרים אישיים שנהוג להעביר בדואר רגיל, והוא מאפשר גם להעביר קבצים ומידע אחר. לדוגמה, באחת החברות שעמה אני עובד קיימת תוכנית ה"מושכת" נתונים ממסד נתונים, מאחסנת אותם בגיליון אלקטרוני ומפיצה אותו לאנשים להם הוא דרוש. כל הפעולות מתבצעות ללא מגע יד אדם. כדי להתמודד עם ביצוע הפעולות באופן ממוכן, דרוש אמצעי שמאפשר לקיים אינטראקציה עם מערכת דואר אלקטרוני מתוך Visual Basic.

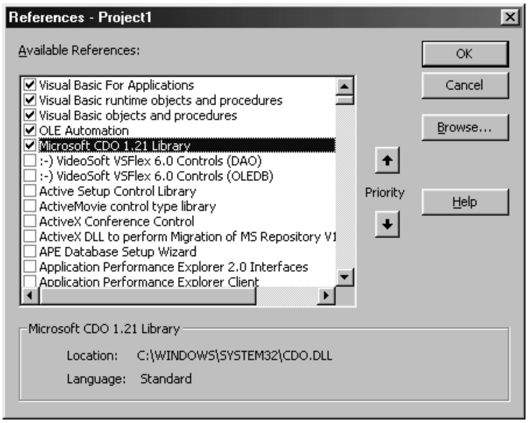
טכנולוגיית Collaboration Data Objects - **CDO** (המכונה גם Microsoft Active Messaging ו- OLE Messaging) היא אמצעי כזה. הטכנולוגיה היא ממשק מונחה אובייקטים המשמש לפנייה ל- Windows Messaging. בעת התקנת גרסאות מסוימות של תוכנת Outlook או תוכנת Exchange, מותקנים בתיקיית המערכת גם קבצים בשם CDO.DLL ו-CDOHTML.DLL.

הערה:

פקדי MAPI ו- MAPI API הם כלים לטיפול בדואר אלקטרוני שנכללו בגרסאות קודמות של Visual Basic. הכלים מוצעים לשימוש גם ב- Visual Basic 6, אך OLE Messaging הוא נוח יותר לשימוש.

כדי שתוכל להשתמש ב-CDO, הוסף לפרויקט הפניה לקובץ CDO.DLL על ידי כך שתבחר את הפריט **Microsoft CDO 1.2 Library** בתיבת הדו-שיח **References** (שאותה תציג על ידי בחירה ב-**Properties**, **References**), כמתואר בתרשים 32.4.

לאחר שתוסיף את ההפניה הקש F2 כדי להציג את Object Browser. Object Browser יאפשר לך לעיין בכל האובייקטים, המאפיינים והשיטות שנוספו לתוכנית על ידי הוספת ההפניה.



תרשים 32.4: שימוש ביכולות CDO, מחייב התקנה של קובץ CDO.DLL במערכת

מה ב-Web

הדוגמאות המובאות בסעיף נוצרו על ידי שימוש בתוכנת Outlook. חומר תיעוד של טכנולוגיית CDO תוכל למצוא באתר Microsoft:

www.microsoft.com/exchange

Microsoft הרחיבה את מיגוון יכולות CDO להעברת מסרים שנכללו ב- Exchange 5.5 הרבה מעבר לנדרש.

הערה:

גירסה CDO קודמת, לפני Exchange 5.0 ו-Outlook נקראה Active Messaging.

כניסה למערכת דואר אלקטרוני

הצעד הראשון בהתחברות למערכת דואר אלקטרוני מ- Visual Basic הוא יצירת Messaging Session. ה**הפעלה** (Session) מייצגת חיבור בין התוכנית שלך (שכתובה ב- Visual Basic) לבין מערכת הדואר האלקטרוני. להתחלת הפעלה חדשה, יהיה עליך ליצור אובייקט Session חדש, ולבצע קריאה לשיטת Logon שלו, כמתואר להלן:

```
Dim objSession As Object
Set objSession = CreateObject("MAPI.Session")
objSession.Logon
```

בעת הפעלה אינטראקטיבית של Exchange או Outlook, ייתכן שיהיה עליך להזין שם משתמש, פרופיל או סיסמה, בהתאם לתצורה שהוגדרה במחשב. בדוגמה שהובאה כאן לא נכללו פרמטרים אלה, כך שכאשר תריץ את קטע הקוד, תונחה להגדיר את הפרמטרים הדרושים לכניסה למערכת.

כדי שתוכל ליצור מערכת ממוכנת לטיפול ב-E-Mail, יהיה עליך למנוע את הצגת תיבות הדו-שיח. עשה זאת על ידי הגדרת פרמטרים נוספים בקריאה לשיטה Logon :

```
objSession.Logon "Your Profile Name",, False, False
```

שני הפרמטרים הראשונים בשורה הם שם וסיסמת הפרופיל. אם הפרופיל שלך אינו דורש הקלדת סיסמה, תוכל להשאיר את פרמטר הסיסמה ריק (כפי שעשינו כאן).

הערה:

ניתן להגדיר פרופיל משתמש על ידי שימוש ביישומון **דואר ופקס** שבלוח הבקרה, או על ידי בחירה בפקודה **Services** מתפריט **Tools** של Outlook.

הפרמטר השלישי, ShowDialog הוא ערך בוליאני, הקובע האם תוצגנה תיבות דו-שיח במהלך הכניסה למערכת הדואר האלקטרוני. אם בפרמטר מוצב ערך True, תוצג הנחיה להזנת פרמטרים, גם אם בשורת הקריאה לשיטה נכללו כל הפרמטרים הדרושים. אם תרצה שתהליך הכניסה למערכת יתבצע באופן אוטומטי לחלוטין, יהיה עליך להציב ערך False בפרמטר זה.

הפרמטר האופציונלי האחרון, NewSession קובע האם בתהליך הכניסה למערכת יוגדר אובייקט Session חדש, גם אם אובייקט כזה כבר קיים. הצבת ערך False תגרום למערכת לחלוק הפעלת E-Mail קיימת עם יישום אחר, כלומר אם Outlook כבר פתוח, המערכת לא תידרש לבצע שוב את תהליך הכניסה.

משלוח הודעה

היתרון העיקרי של טכנולוגיית CDO הוא שהיא משתמשת בתחביר הרגיל של עבודה עם אוספים (Collections) לייצוג Messages ואובייקטים אחרים. באוספים כבר דנו בפרקים אחרים, כך שסביר להניח שהפעולות הדרושות להוספה ולגריעת אובייקטים לאוסף כבר מוכרות לך. בדומה לאוספים אחרים, גם במקרה זה ניתן להשתמש בשיטה Add של האוסף ליצירת אובייקטים באחד משני אופנים : על ידי החזרת הפניה לאובייקט, או על ידי שימוש בפרמטרים שיגדירו את מאפייני האובייקט.

ראה: "יצירת מחלקות המכילות אובייקטים", פרק 16.

למשלוח הודעה על ידי שימוש ב-CDO בצע את הפעולות הבאות :

1. הוסף אובייקט Message חדש לאוסף Messages של תיבת Inbox.
2. הגדר את הטקסט והנושא של Message וצרף קבצים לאוסף Attachments שלו.
3. צרף שמות (ו/או כתובות) נמענים לאוסף Recipients.
4. פענח שמות נמענים כדי לזהות את הכתובות הממשיות שלהם.
5. קרא לשיטה Send כדי לשלוח את ההודעה.

הקוד שיבצע את הפעולות הוא קצר וקל להבנה:

```
'...Assume we have already logged on

Dim objMessage as Mapi.Message
Dim objSession as Mapi.Session

'Add a new message to the Inbox
Set objMessage = obj.Session.Inbox.Messages.Add

'Set the subject and body text
objMessage.Subject = "Test Message"
objMessage.Text = "This is a test, repeat only a test message!"

'Add an attachment
objMessage.Attachments.Add "Monthly Report",,
        ActMsgFileData,"C:\MyReport.XLS"

'Add a recipient to the list
objMessage.Recipients.Add "bsiler@bigfoot.com",, ActMsgto
objMessage.Recipients.Resolve

'Send The Message
objMessage.Send
```

בקטע הקוד ראית שטכנולוגיית CDO משתמשת באוספים לניהול האובייקטים Message, Recipient ו-Attachment. קטע הקוד נפתח ביצירת אובייקט Message חדש ואחר מתבצעת קריאה לשיטה Send לצורך משלוח Message. השיטות והמאפיינים של אובייקטים אלה מוגדרים בקובץ CDO.DLL וניתן לעיין בהם באמצעות Object Browser.

הפעולה היחידה בקטע הקוד, שעשויה להיות לא ברורה היא נושא פענוח שמות משתמשים. במערכת הדואר האלקטרוני, בפנקס כתובות (Address Book), פעולת הפענוח (Resolution) היא פעולה פשוטה שכרוכה בנטילת שם "ידידותי" שמופיע בפנקס הכתובות (כגון Dad At Work) והמרתו לכתובת שמשמשת את שרת הדואר האלקטרוני (כגון SMTP: jsmith@somewhere.com).

גישה לתוכן הודעה

זה עתה למדת כיצד לשלוח הודעת דואר אלקטרוני, על ידי שימוש ב-CDO. תוכל להשתמש באוסף Messages לגישה להודעות שטרם נקראו (או להודעות שנקראו) תוך שימוש בקוד Visual Basic. אחד השימושים האפשריים של יכולת זו הוא: משלוח הודעת דואר אלקטרוני עם נושא מסוים, ואחר שימוש בתוכנית שתענה באופן אוטומטי, או תאחסן את התגובות במסד נתונים. נתייחס כעת לדוגמת הקוד הבאה:

```
For Each objMessage In objSession.Inbox.Messages
   If objMessage.Unread = True Then
     If Instr(objMessage.Subject, "Catalog Request") Then
         ProcessMessage objMessage.Text
     End If
  End If
Next objMessage
```

קטע הקוד נעזר בלולאת For Each לביצוע מעבר על הפריטים שבאוסף Messages. אם הודעה כלשהי טרם נקראה (ואת זאת ניתן לבדוק לפי ערך המאפיין Unread שלה) ושורת הנושא שלה מכילה ביטוי מסוים, מתבצעת קריאה לפונקציה ProcessMessage, שהוגדרה על ידי המשתמש. הפונקציה ProcessMessage, שמקבלת את הטקסט ההודעה כפרמטר, יכולה לבצע מיגוון פעולות על ההודעה, ביניהן מענה או הוספת תוכן ההודעה לטבלה במסד נתונים.

קל לעבוד עם הודעות דואר אלקטרוני, מפני שניתן להתייחס לכל הודעה כאל אובייקט עצמאי שיש לו שיטות ומאפיינים. ניתן להיעזר בלולאות For...Each כדי לעבור על הפריטים באוסף ולבצע עליהם פעולות רצויות. קטע הקוד הבא ידפיס את תוכן שורת הנושא של כל הודעה שטרם נקראה, ונשלחה לאחר 7/4/1998:

```
For Each objMessage In objSession.Inbox.Messages
        If objMessage.Unread = True And objMessage.Submitted = True Then
                If objMessage.TimeSent >= "7/4/98" Then
                        Debug.Print objMessage.Sender & objMessage.Subject
                End If
        End If
Next objMessage
```

קטע הקוד נעזר בסדרת משפטי If שבודקים האם ההודעה הפעילה טרם נקראה, האם היא הועברה לטיפול והאם היא נשלחה לאחר 7/4/98.

טיפ:

שימוש בממשק CDO עלול להפוך את התוכנית למעט פגיעה יותר לשגיאות, משום שהיא תאבד את השליטה על אינטראקציה עם המשתמש. מסיבה זו אני ממליץ שתכתוב פונקציה לביצוע כל אחת מהפעולות הדרושות (כגון כניסה למערכת, משלוח הודעה וכדומה) אשר תשלב קוד לאיתור מצבי שגיאה וטיפול בהם.

בסעיף זה "העפנו מבט" על ספריית CDO. אם תרצה להציג את רשימת השיטות והמאפיינים המוצעים על ידי הספריה תוכל להשתמש ב- Object Browser או לעיין בקובץ העזרה CDO.CHM.

פקד Internet Transfer

פקד Internet Transfer הוא כלי נוסף לעבודה בסביבת Web ב- Visual Basic. הפקד מאפשר להתקשר למחשב אחר לצורך העברת קבצים. הפקד מהווה ניסיון להקל את השימוש בשני פרוטוקולים פופולריים: HTTP ו-FTP. פרוטוקול HTTP הוא ה"שפה" שבה דפדפני Web מתקשרים עם שרתי Web. אם כבר התנסית בהעברת מידע באמצעות ה-Web, סביר להניח שזכית להכיר את פרוטוקול FTP (File Transfer Protocol). בסעיף זה נציג דוגמאות לשימוש בשני הפרוטוקולים.

אחזור HTML

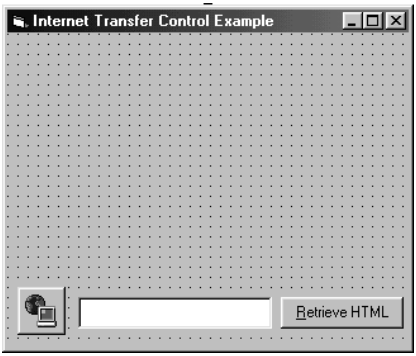
בגישה לדף Web, הדפדפן נעזר בפקודה GET (מוגדרת בפרוטוקול HTTP) כדי לבקש משרת Web מסמך. למשל, כשתקליד כתובת http://www.food.com/peanuts.html, הדפדפן יתקשר למחשב שכתובתו www.food.com ויעביר אליו את הפקודה GET /peanuts.html. קובץ HTML יוחזר לדפדפן במבנה טקסט פשוט והדפדפן ישתמש ברכיבים הלוגיים שלו כדי לעצב את הטקסט באופן הרצוי.

בעת שימוש בפקד Internet Transfer בשילוב עם פרוטוקול HTTP תוכל "למשוך" קוד HTML שאותו תוכל לעבד בתוכנית שלך. לדוגמה, מדי פרק זמן מסוים תוכל להעביר לאתר Web דרישה לקבלת נתונים על מנייה מסוימת. התוכנית שלך תוכל לשמור את הנתונים שיתקבלו במסד נתונים של SQL, שיאפשר למשתמשים נוספים להציג נתונים אלה. כך שניתן לומר שפקד Internet Transfer יכול לשמש לאיסוף נתונים אוטומטי מאתרי Web.

כדוגמה, נציג שימוש בפקד לאחזור קוד HTML שיוצג בטופס. צור את פרויקט הדוגמה על ידי ביצוע הפעולות הבאות:

1. פתח את Visual Basic וצור פרויקט חדש מסוג Standard EXE.
2. הצג את תיבת הדו-שיח Project Components והוסף לפרויקט הפניה ל- Internet Transfer Control 6.0.
3. שרטט מופע של הפקד על הטופס. הפקד יוצג כסמל, ללא תלות בגודל בו תשרטט אותו.
4. הוסף לטופס תיבת טקסט וקרא לה txtURL ולחצן פקודה בשם cmdRetrieve.

לאחר שתשלים את ביצוע הפעולות, הטופס שלך ייראה כמו זה שמוצג בתרשים 32.5, ותהיה מוכן להוסיף לפרויקט הדוגמה קוד.



תרשים 32.5: פקד Internet Transfer אינו מוצג באופן אוטומטי בארגז הכלים

פקד Internet Transfer תומך באירוע אחד, StateChanged. מטרת האירוע הוא להודיע לתוכנית על התרחשות פעילויות שונות. לדוגמה, הפקד מצוי במצב אחד בעת התחברות לשרת Web, ובמצב אחר בעת אחזור קוד HTML. המצב הפעיל של הפקד מאוחסן בפרמטר State של שגרת האירוע. הזן כעת את הקוד לאירוע StateChanged של התוכנית, אשר מובא בתוכנית 32.3.

תוכנית 32.3: INETDEMO.VBP - אחזור HTML גולמי משרת Web.

```
Private Sub Inet1_StateChanged(ByVal State As Integer)

 Select Case State

   Case icResponseCompleted
      sTemp = Inet1.GetChunk(100)
      Do While sTemp <> ""
        Me.Print sTemp;
        sTemp = Inet1.GetChunk(100)
      Loop
      Me.Print
  Case icError
     MsgBox Inet1.ResponseInfo, vbCritical, "ERROR!"
 End Select
End Sub
```

קטע הקוד מתייחס לשני מצבים בלבד, icResponseCompleted ו-icError. מצב icResponseCompleted מציין שהפקד סיים לקבל מענה על בקשה. השיטה GetChunk משמשת לאחזור הטקסט מהמאגר (Buffer) ביחידות של 100 בתים בכל פעם והדפסת הטקסט שאוחזר על הטופס.

הערה:

רשימה מלאה של המצבים שבהם תומך הפקד ושל משמעויות כל מצב תוכל למצוא תחת המושג StateChanged Event בעזרה המקוונת.

קוד שגרת האירוע שהובאה בדוגמה הוא אומנם המרכיב העיקרי בתוכנית הדוגמה, אך עדיין יהיה עליך להוסיף קוד שיטפל באירוע Click של הלחצן cmdRetrieve שייזום את העברת הבקשה. קטע הקוד בתוכנית 32.4 ייעזר בשיטה Execute של הפקד, כדי להעביר את פקודת פרוטוקול HTTP:

תוכנית 32.4: INETDEMO.VBP - השיטה Execute.

```
Private Sub cmdRetrieve_Click( )
    Inet1.Protocol = icHTTP
    Inet1.Execute CStr(txtURL), "GET /"

    Do While Inet1.StillExecuting
        DoEvents
    Loop
    MsgBox "Done!"
End Sub
```

לאחר שתזין את הקוד, הרץ את הפרויקט והזן בתיבת הטקסט כתובת URL. לחץ על הלחצן, וטקסט HTML יוצג על הטופס כמתואר בתרשים 32.6.



תרשים 32.6: פקד Internet Transfer יכול לשמש לאחזור קוד HTML מאתר Web

בתוכנית הדוגמה הצגנו את תוצאות הפעולה על הטופס. כדי לעבוד עם קוד HTML כמחרוזת, אחסן אותו במשתנה מחרוזת או בפקד תווית. כאשר תסתיים פעולת אחזור קוד HTML מהאתר, תוכל להיעזר בפונקציות סטנדרטיות לטיפול במחרוזות כדי לחפש את המידע הרצוי בתוך קוד HTML.

העברת קבצים

פרוטוקול FTP, שהוא פרוטוקול העברת הקבצים המשמש ב-Web, הוא פרוטוקול ותיק יותר מפרוטוקול HTTP. בדומה לעבודה ב-Web, גם העבודה בפרוטוקול FTP מתבצעת תוך שימוש במחשב שרת ובמחשב לקוח. אומנם בשנים האחרונות מוצעות תוכנות לקוח בעלות ממשק גרפי לפרוטוקול FTP, אך תוכנות הלקוח הסטנדרטיות של פרוטוקול זה עדיין כוללות ממשק טקסטואלי.

אם השתמשת בעבר בשורת הפקודה של DOS, סביר להניח שתצליח להבין את אופן פעולת FTP. FTP מאפשר לשנות את התיקיה הפעילה, להציג רשימת קבצים ולהעתיק קבצים, בדיוק באותם אופני ביצוע בשורת הפקודה של DOS. ההבדלים היחידים בין הפעולות הם שכעת אתה מבצע את פעולות החיפוש על ספריות המצויות במחשב מרוחק ומשתמש בפקודות שונות מעט. הפקודות get ו-putFTP משמשות לאחזור קבצים מהמחשב המרוחק, ולמשלוח קבצים אליו, בהתאמה.

משום שפרוטוקול FTP משמש לעבודה עם מיגוון פלטפורמות רחב (UNIX, Windows, AS/400), הוא מהווה אמצעי אידיאלי להעברת נתונים בין מחשבים שונים. פקד Internet Transfer מהווה אמצעי נוח לביצוע פעולות אלו מתוך Visual Basic. בסעיף זה נדגים יצירת תוכנית פשוטה לאחזור קבצים משרת FTP.

נתחיל על ידי כך שנתבסס על הפרויקט שיצרנו בסעיף הקודם. צור את פרויקט הדוגמה שתואר בסעיף זה והוסף לו לחצן פקודה נוסף, שתקרא לו cmdGetFile. הקוד בשגרת האירוע Click של הלחצן מובא בתוכנית 32.5.

תוכנית 32.5: INETDEMO.VBP שימוש ב-FTP בעזרת Internet Transfer Control.

```
Private Sub cmdGetFile_Click( )
    Me.Cls
    Inet1.Protocol = icFTP
    Inet1.UserName = "anonymous"
    Inet1.Password = "guest@"
    Inet1.Execute "ftp.microsoft.com", "pwd"
    Do While Inet1.StillExecuting
        DoEvents
    Loop
    Inet1.Execute, " dir *.txt"
    While Inet1.StillExecuting
        DoEvents
    Wend
    Inet1.Execute, "GET dirmap.txt c:\dirmap.txt"
    Do While Inet1.StillExecuting
        DoEvents
    Loop
    MsgBox "Done!"
End Sub
```

קטע הקוד נעזר בפקד Internet Transfer לשימוש בשלוש פקודות בפרוטוקול FTP :

- pwd - מציגה את התיקיה הפעילה במחשב המרוחק.
- dir - מציגה את רשימת הקבצים המאוחסנים בתיקיה הפעילה במחשב המרוחק.
- get - מאחזרת קובץ מהמחשב המרוחק למחשב הלקוח.

לאחר כל פקודה השתמשנו בלולאת While כדי לוודא שהפקד סיים לבצע אותה, לפני ביצוע הפקודה הבאה. הקוד בשגרת האירוע StateChanged גורם לתגובות שתתקבלנה מפקודות pwd ו-dir להופיע בטופס. הפקודה האחרונה get, תעביר קובץ משרת FTP למחשב המקומי. תוכל להיעזר בדוגמה לביצוע אחזור קבצים ממוכן ממחשב מרוחק, למשיכת מפות מזג אוויר עדכניות, או נתונים דומים מאתרי Web.

מכאן...

בפרק התייחסנו למספר נושאים שקשורים לאינטרנט אך אינם בתחום העיסוק הרגיל ב-Web. המקום הטוב ביותר ללמוד על ה-Web, הוא כמובן ה-Web עצמו. מידע נוסף על נושאים הקשורים ב- Visual Basic, תוכל למצוא בפרקים הבאים :

- על יסודות השימוש ב-API תוכל ללמוד בפרק 20 **גישה ל-API של Windows**.
- מידע נוסף על קבצי INI וקבצי טקסט תוכל למצוא בפרק 21 **עבודה עם קבצים**.
- מידע נוסף על ממשקים נוספים של טכנולוגיית OLE, כדוגמת CDO, תוכל למצוא בפרק 22 **שימוש בפקד OLE לשליטה על יישומים אחרים**.

חלק 8

נספחים


New Project
Microsoft Visual Basic
New
Existing
Recent
נספח 1: הכרת סביבת הפיתוח
נספח 2: אריזת היישום
נספח 3: תקציר פקודות SQL
נספח 4: דפי שרת פעילים
נספח 5: התקליטור המצורף
אינדקס עברי
אינדקס לועזי
Open
Cancel
Help
Don't show this dialog in the future

נספח 1

הכרת סביבת הפיתוח

מה בפרק?

- הבנת הרכיבים העיקריים בסביבת הפיתוח
- התחלה
- אזור העבודה של Visual Basic

לסביבת הפיתוח המשולבת (IDE) המשמשת כממשק Visual Basic מספר כלים שבאמצעותם תוכל לפתח את היישום. בין גרסאות 4 ל-5 עברה IDE שינויים משמעותיים, אולם הממשק מאוד דומה לקודמו. בנספח זה נעשה סיור מהיר ונלמד יותר על הממשק ומרכיביו.

הבנת הרכיבים העיקריים בסביבת הפיתוח

מגירסה Visual Basic 5 תומכת בסביבת **ממשק מרובה מסמכים** (MDI). אם המושג MDI אינו מוכר לך, דמה אותו לפתיחת מסמכים רבים בתוכנת WORD או EXCEL. כל מסמך נפתח בחלון משני בחלון הראשי. בדומה ליישומי MDI אחרים, תוכל לבחור בחלון המשני ולהרחיבו על כל שולחן העבודה או להציג מספר חלונות בו-זמנית.

טיפ:

אם אינך משתמש עדיין במסך גדול (17" ומעלה) וברזולוציה גבוהה, מומלץ כי תעשה זאת. הינך זקוק לשטח מסך גדול כדי להשתמש בממשק בצורה יעילה. אני ממליץ על מסך ברזולוציה 600X800 לכל הפחות או 768X1024 אם כרטיס המסך והעיניים שלך מאפשרים זאת.

ניתן לערוך מספר פרויקטים במופע יחיד של Visual Basic. אין צורך לסגור פרויקט כדי לפתוח ולבצע שינויים בפרויקט אחר. הדבר נוח בפיתוח פרויקטים המקושרים ביניהם היות וניתן ולשמור ולהדר כל פרויקט כקבוצה בבחירת תפריט יחידה.

עוד רכיב חדש יחסית הוא היכולת לעגן כל חלון או סרגל כלים. כיום סרגל כלים או חלון יכול לצוף באמצע המסך או להופיע לאורך אחד הקצוות. חלונות מעוגנים מכילים סרגלי כלים ושורות כותרת, בדומה לארגז הכלים.

בנוסף לאפשרויות ניהול החלונות, Visual Basic מציעה מספר כלים שימושיים להכנסת קוד בצורה קלה יותר. הכינוי שניתן להם על ידי Microsoft הוא **AutoListMember** (חבר ברשימה אוטומטית) ו-**AutoQuiqInfo** (מידע אוטומטי מהיר). אתה פשוט תקרא להם נפלאים. אם אתה מתקשה לזכור את קבועי MessageBox, מאפייני הפקד או אפילו פרמטרים בפונקציה, עורך הקוד ייסייע לך להשלים את המידע בצורה אוטומטית. בזמן ההקלדה, ברגע שתקיש רווח לאחר קריאה לפונקציה MsgBox, הפרמטרים יוצגו בצורת טיפ מתוך רשימה נפתחת של קבועים קיימים.

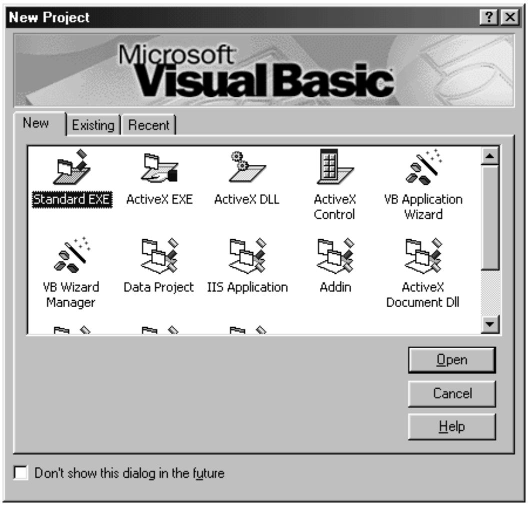
טיפ:

כאשר מוצגת הרשימה הנפתחת, אינך חייב להשתמש בעכבר כדי לבחור פריט. פשוט המשך להקליד (או השתמש במקשי החיצים) עד אשר הפריט שברשימה יסומן. אז הקש על אחד מהמקשים: רווח, פסיק (,), Enter או Tab והמשך בתוכנית.

התחלה

כאשר מפעילים את Visual Basic מופיעה תיבת דו שיח New Project, כמתואר בתרשים נספח 1.1. לתיבת דו-שיח זו שלוש כרטיסיות עיקריות:

- New - מאפשרת לבחור סוג פרויקט מבין הסוגים המוצעים.
- Existing - מאפשרת חיפוש פרויקט שכבר נוצר ונשמר.
- Recent - רשימת פרויקטים שנפתחו לאחרונה. האחרונים מוצגים ראשונים.



תרשים נספח 1.1: הכרטיסיה New בתיבת הדו-שיח New Project

הערה:

קיימת גירסה נוספת של תיבת הדו שיח New Project. גירסה זו מופיעה כאשר Visual Basic כבר מופעלת ובוחרים **File**, **New Project** מתוך התפריט. גירסה זו מוצגת בתרשים נספח 1.1 אבל אינה כוללת את הכרטיסיות Existing ו-Recent.

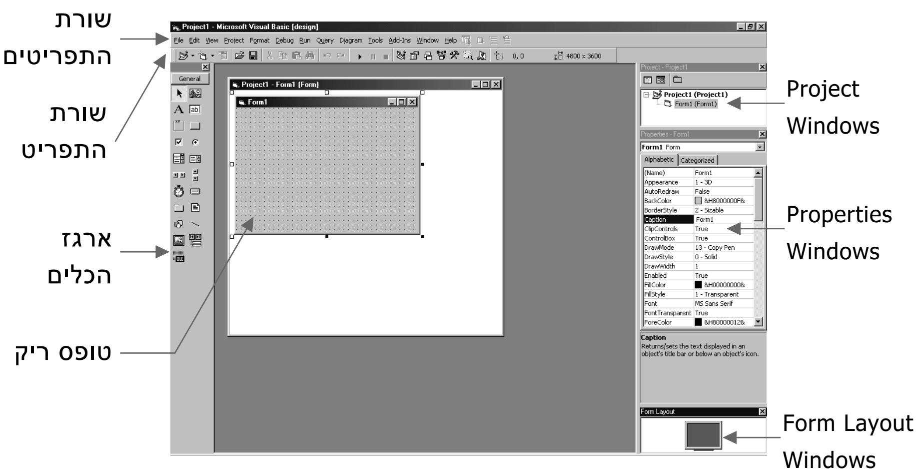
אם תבחר ליצור פרויקט חדש, Visual Basic תיצור תבנית מתאימה המבוססת על הבחירה . ניתן לבחור באחד מסוגי הפרויקטים הבאים:

- **Standard EXE** השתמש בסוג זה ליצירת יישום סטנדרטי של Windows (קובץ EXE). סביר להניח כי תשתמש בסוג זה לעיתים תכופות.
- **ActiveX EXE** שרת אוטומטי זה מבצע משימות כחלק מיישום רב-שכבות. התוצאה הסופית היא יישום המכיל מחלקות ציבוריות הנגישות מתוכניות אחרות או פועלות בצורה עצמאית. הוא נקרא קודם OLE Automation Server.
- **ActiveX DLL** תוכנת אוטומציה לשליטה מרחוק נוצרת כקובץ DLL. קבצים אלה אינם יכולים לפעול עצמאית, אולם היות והם מופעלים מתוך תהליך, הם מהירים יותר מאשר הפעלה מתהליך חיצוני (ActiveX EXE).

- **ActiveX Control** באמצעות אפשרות זאת ניתן ליצור פקדים מותאמים (OCX). אפשר להשתמש בהם בתוכניות Visual Basic או בתוכניות תומכות ActiveX.
- **Visual Basic Application Wizard** אפשרות זו תבנה שלד יישום (בדומה לתבנית במעבד תמלילים). לאחר מכן ניתן להתאים את היישום לצרכי התוכנית
- **Data Project** סוג זה מכיל מעטפת יישום התומך בנתונים. ניתן להתאים את הרכיבים כך שפיתוח תוכנית התומכת במאגרי נתונים ייעשה בצורה מהירה.
- **IIS Application** אפשרות זו יוצרת פרויקט הניתן לשילוב עם שרת המידע ל- Windows NT Web (IIS) וניתן להפעילו בסביבה מבוססת WEB (דורש PWS או IIS מותקן).
- **Add-in** סוג זה מוסיף רכיבים שימושיים ל- Visual Basic. דוגמה טובה לתוספת מסוג זה היא Visual Data Manager.
- **ActiveX Document DLL** סוג זה יוצר קובץDLL הניתן לשימוש בשילוב עם Microsoft Internet Explorer.
- **ActiveX Document EXE** יוצר יישום הניתן להפעלה מ- Internet Explorer.
- **DHTML Application** אפשרות זו יוצרת מסגרת עבודה של HTML דינמי הניתן להפעלה מתוך דפדפן.

אזור העבודה של Visual Basic

לאחר שבחרת בסוג הפרויקט, תוצג סביבת הפיתוח. כאן למעשה מתבצעת העבודה על יצירת המופת. סביבת הפיתוח הבסיסית מוצגת בתרשים נספח 1.2. סביר להניח כי Visual Basic נראתה כך בהפעלה הראשונה.



תרשים נספח 1.2: שולחן העבודה של Visual Basic מספק מיגוון כלים שבאמצעותם אפשר ליצור תוכניות

כפי שאפשר לראות, ל- Visual Basic ולישומי Windows אחרים הרבה אלמנטים משותפים. סרגלי הכלים והתפריטים דומים לאלה של Office. מספר תפריטים נראים זהים : **File**, **Edit**, **Help** ואחרים.

שורת התפריטים

מפתחים רבים מעוניינים בקיצורי דרך לביצוע פעולות שכיחות. כמו בתוכניות Windows אחרות, התפריטים בחלק העליון של סביבת העבודה יכולים להיפתח על ידי הקשה בו-זמנית על Alt והמקש המתאים לאות המסומנת בקו תחתי. לאחר שהתפריט נפתח, אפשר פשוט להקיש על המקש המתאים לאות המסומנת בפקודה הרצויה. לדוגמה, הקש Alt+F כדי לפתוח את התפריט **File** ואחר הקש P כדי לבחור את הפקודה **Print**.

Visual Basic גם מציעה מספר קיצורי דרך המאפשרים לעקוף את התפריט. רובם מוצגים לצד הפריט המתאים. לדוגמה, בתפריט **View** אפשר להבחין מימין לפריט Object Browser בסימון F2. הכוונה היא שניתן להפעיל את Object Browser על ידי הקשה על F2. בטבלת נספח 1.1 מופיעים מספר קיצורים מהירים, אך היכרות אישית עם התפריטים יכולה להועיל.

טבלת נספח 1.1: מקשי קיצור

פריט בתפריט	קיצור	תאור
Edit, Cut	Ctrl+X	מוחק את הטקסט או הפקד המסומן מהמיקום הקיים ומעביר אותו לתצוגת הלוח
Edit, Copy	Ctrl+C	יוצר העתק של הטקסט או הפקד הנבחר ללוח, אבל אינו מוחק אותו מהמיקום המקורי
Edit, Paste	Ctrl+V	מדביק את תוכן הלוח לטופס הפעיל או לחלון הקוד
Edit, Undo	Ctrl+Z	מבטל את השינוי האחרון
Edit, Find	Ctrl+F	מוצא קטע טקסט (עליך להיות בחלון העריכה לביצוע פקודה זו)
File, Open	Ctrl+O	פותח פרויקט
File, Save	Ctrl+S	שומר את הקובץ הנוכחי (לא את הפרויקט)
File, Print	Ctrl+P	מציג את תיבת הדו-שיח Print, כדי להדפיס טופס, מודול נוכחי או את כל היישום
View, Project Explorer	Ctrl+R	מציג את Project Explorer (אם אינו מוצג)
View, Properties Window	F4	מציג את חלון Properties Window

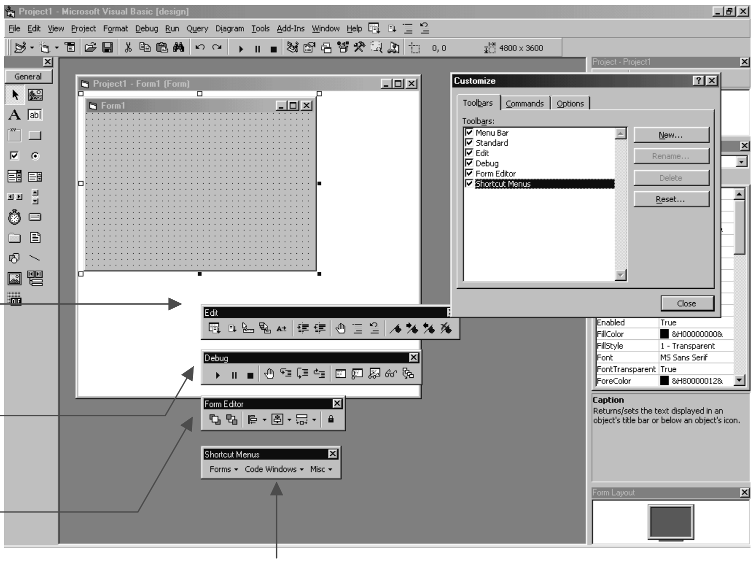
טיפ:

מומלץ שתנסה את הקיצורים. בנוסף לרשימה הקודמת ל- Visual Basic מספר טריקים לא כל כך ברורים מאליהם כמו Ctrl+Y (מוחק שורה קוד שלמה ומעביר אותה לתצוגת הלוח), שנראה כאות הכרה למעבד התמלילים WordStar.

גישה לפונקציות מתוך שורת התפריטים

שורת התפריט של Visual Basic מאפשרת גישה למספר פונקציות שכיחות. ארבעה סרגלי כלים זמינים (ראה תרשים נספח 1.3):

- **Standard**. סרגל הכלים Standard מוצג כברירת מחדל ומאפשר גישה מהירה לפונקציות שכיחות.
- **Debug**. לסרגל הכלים Debug יש ארבעה לחצנים שימושיים לניפוי שגיאות בתוכנית.
- **Edit**. לחצני סרגל הכלים Edit יעילים בזמן כתיבת קוד.
- **Form Editor**. לסרגל הכלים Form Editor לחצנים שיעזרו לשפר את מראה הפקדים.



תרשים נספח 1.3: סרגל הכלים הסטנדרטי מעוגן מתחת לשורת התפריטים. שאר הסרגלים צפים על שולחן העבודה. ניתן לשנות סרגל על ידי תיבת הדו שיח Customize

סרגל הכלים הסטנדרטי הוא היחיד שמוצג בהפעלה הראשונה. ניתן לבחור איזה סרגל כלים יוצג על ידי בחירה (Select) ב-**View**, **Toolbars** או פשוט על ידי לחיצה ימנית על סרגל כלים קיים. כל אחד מהם יכול לצוף בצורה חופשית או להיות מעוגן מתחת

לשורת התפריטים. בהפעלת התוכנית, מיקומם יהיה מיקומם האחרון בעת הסגירה הקודמת. לחיצה על **View**, **Toolbars**, **Customize** מאפשרת לשנות תפריט קיים או אף ליצור חדש.

סרגלי הכלים של Visual Basic תואמים לסטנדרטים של הדור האחרון בתכנות, במובן זה שהם מכילים תיאור כלי. תיאור כלי הוא תיבה צהובה קטנה המופיעה כאשר משאירים את סמן העכבר מספר שניות על לחצן מסוים, ומכילה תיאור של הלחצן המסומן.

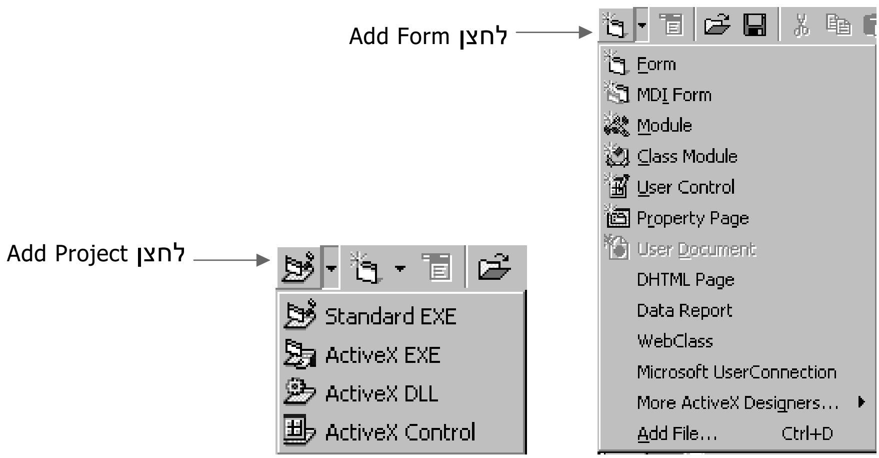
טיפ:



תיאור כלי יכול גם להציג ערך משתנה בתוך חלון קוד. השתמש באפשרות זו על ידי השארת סמן העכבר על שם משתנה בזמן שהרצת הקוד מופסקת. אפשרות זו יכולה לחסוך זמן רב אם אתה רגיל להשתמש ב- Watch Window או להדפיס ערכים ב- Immediate Window בזמן ההפעלה.

הצג את כל סרגלי הכלים כמוסבר בפיסקה הקודמת, והעבר את סמן העכבר מעל הפקודות כדי להכיר אותן. זכור שתמיד תוכל להשתמש בתיאור הכלים אם אינך בטוח מה מייצג כל לחצן.

שני לחצנים מצריכים תשומת לב מיוחדת. הלחצנים Add Project ו- Add Form גורמים להופעת תיבות רשימה נפתחות (ראה תרשים נספח 1.4). אם תבחר פריט, הלחצן הראשי ישנה את סוגו בהתאם לפריט שבחרת.



תרשים נספח 1.4: רשימות נפתחות מאפשרות לציין איזה סוג פרויקט ברצונך לבחור

הלחצן Add Project מאפשר להוסיף פרויקט לשולחן העבודה. ניתן לבחור באחד מהסוגים הבאים:

- Standard EXE
- ActiveX EXE
- ActiveX DLL
- ActiveX Control

הלחצן Add Form מאפשר להוסיף אחד מתוך הפריטים הבאים לפרויקט:

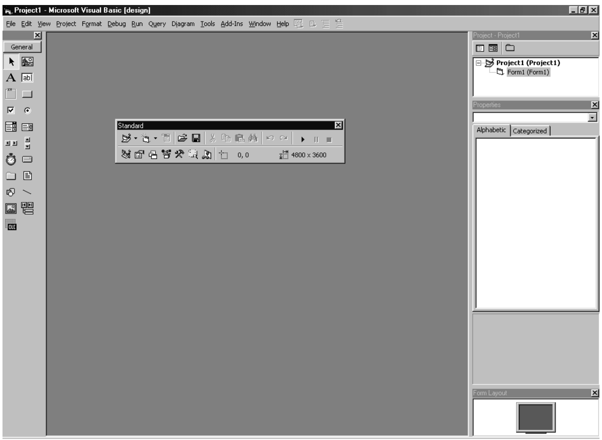
- Form
- MDI form
- Module
- Class module
- User control
- Property page
- Existing files

מאוחר יותר תלמד עוד על מרבית הלחצנים. אולם שני אזורים בסרגל הכלים ראויים להתייחסות. בקצה הימני יש שני קטעים, כל אחד מכיל זוג מספרים. שני קטעים אלה מציגים את המיקום ואת גודל הטופס או הפקד הנוכחי. שני המספרים בקטע הראשון מציגים את המיקום האופקי והאנכי בהתאמה, הנמדדים מהפינה השמאלית העליונה של המסך עד לפינה השמאלית העליונה של הטופס (בעבודה על טופס) או עד לטופס אליו שייך הפקד. שני המספרים בקטע השני מציינים גודל אופקי ואנכי בהתאמה, של האובייקט הנוכחי. מספרים אלה אינם נראים כאשר עובדים בחלון עריכת הקוד.

הערה:

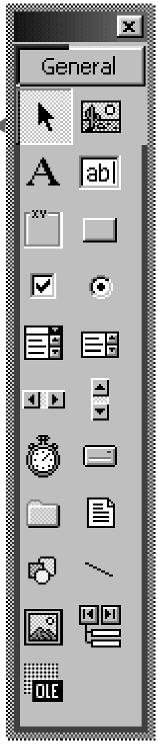
המספרים המציינים גודל ומיקום ניתנים ביחידות טוויפ (twip). טוויפ הינה יחידת מדידה בה משתמשת Visual Basic כדי להבטיח שיחידות המדידה גודל והמיקום תהיינה זהות על מסכים בגדלים שונים. טוויפ שווה ל 1/20 נקודת הדפסה. 1440 טוויפ מהווים שטח אינץ' לוגי (כמות השטח הנדרשת על המסך להדפסת אינץ' אחד במדפסת).

הערה אחרונה על סרגלי כלים, אם אינך אוהב אותם ממוקמים בחלק העליון של המסך ניתן להזיז אותם על ידי לחיצה על הקווים הכפולים שבקצה בשמאלי וגרירה למקום חדש. ניתן לעגן אותם בכל קצה של המסך, או להשאיר אותם צפים במרכז המסך, כמתואר בתרשים נספח 1.5.



תרשים נספח 1.5: אפילו סרגל הכלים הסטנדרטי יכול לצוף על שולחן העבודה

ארגון פקדי Visual Basic



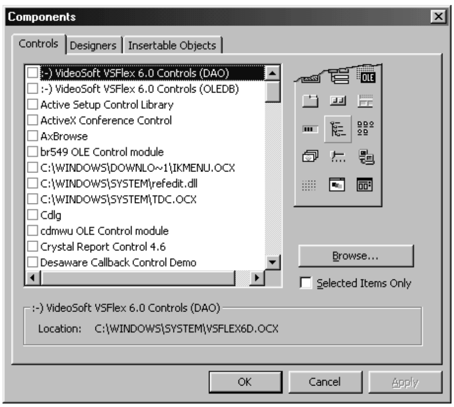
הפקדים הם הלב והנפש של התוכניות שתיצור. הם מאפשרים להוסיף פונקציונליות לתוכנית בצורה קלה ומהירה. אפשר למצוא פקדים שיאפשרו לערוך טקסט, להתחבר לבסיס נתונים, לקבל קבצי נתונים ממשתמש, או להציג ולערוך תמונות.

מצביע (Pointer)

ברור שמספר כה רב של פקדים צריך לשמור בצורה מאורגנת. זה תפקיד ארגז הכלים (ראה תרשים נספח 1.6). ארגז כלים זה מכיל לחצנים המייצגים את הפקדים הנגישים לשימוש בתוכנית (רשימת הפקדים הבסיסים מופיעה בטבלת נספח 1.2). לחיצה על אחד מפקדי ה"כלים" מאפשרת לצייר פקד מאותו סוג בטופס. לחיצה כפולה ממקמת פקד בעל גודל המוגדר כברירת מחדל, במרכז הטופס הנכוחי. לחיצה על כלי המצביע (Pointer) שבפינה השמאלית העליונה של ארגז הכלים מבטלת פעולת יצירת פקד ומחזירה את סמן העכבר למצב עבודה רגיל.

תרשים נספח 1.6: מערכת הפקדים הבסיסית נגישה בהפעלה הראשונה. ניתן להזיז את ארגז הכלים על המסך

אפשר להוסיף פקדים לארגז הכלים על ידי בחירה ב-Project, Components. פעולה זו פותחת את תיבת הדו-שיח Components (ראה תרשים נספח 1.7). בתיבת דו-שיח זו אפשר לבחור כל פקד נוסף שהותקן במערכת. אם תבחר להוסיף פקדים לארגז הכלים הם יופיעו בו לאחר שתלחץ על OK או על לחצן Apply.



תרשים נספח 1.7: הוספת פקדים באמצעות תיבת הדו-שיח Components

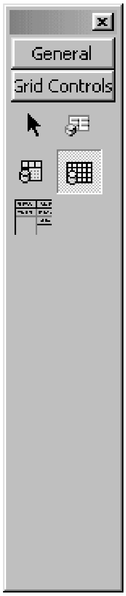
טיפ:

ניתן לגשת לתיבת הדו-שיח Components על ידי לחיצה ימנית על ארגז הכלים ובחירת הפריט Components מתוך הרשימה הנפתחת, או באמצעות הקשת Ctrl+T.

טבלת נספח 1.2: הפקדים הבסיסיים של Visual Basic

שם הפקד	**פעולה**
PictureBox	מציג תרשים גרפי
Label	מציג טקסט שהמשתמש אינו יכול לשנות
TextBox	מציג טקסט שהמשתמש יכול לערוך
Frame	מספק שיטה לארגון פקדים
CommandButton	מאפשר למשתמש להפעיל פעולת תוכנית יכול לשלב סמל, שם ותיאור כלי
CheckBox	מציג או מאפשר בחירה מתוך שני מצבים: כן\לא או נכון\לא נכון
OptionButton	מציג ומאפשר בחירת פריט אחד מרבים (ידוע גם כלחצן רדיו)
ComboBox	מאפשר בחירת פריט מרשימה או הוספת פריט חדש לרשימה
ListBox	מציג רשימת פריטים מתוכה אפשר לבחור פריט אחד או יותר
HscrollBar	יוצר ערך מספרי המבוסס על המיקום האופקי של פס הגלילה
VscrollBar	זהה לקודם אבל אנכי. פסי הגלילה עובדים כמו ב-Windows
Timer	מבצע פעולה לאחר שעבר פרק זמן מסוים
DriveListBox	מציג ומאפשר בחירת כונן קיים במחשב
Dir ListBox	מציג ומאפשר בחירת תיקייה בכונן או החלפת כונן
FileListBox	מציג ומאפשר בחירת קבצים מתוך תיקיה
Shape	מציג צורות גיאומטריות בטופס
Line	מציג קווים על הטופס
Image	מציג תמונה גרפית. ההופעה דומה לפקד Picture אבל הפעולה שונה
Data	מספק קישור למאגר נתונים
OLE	מספק דרך להתקשר לשרתי OLE

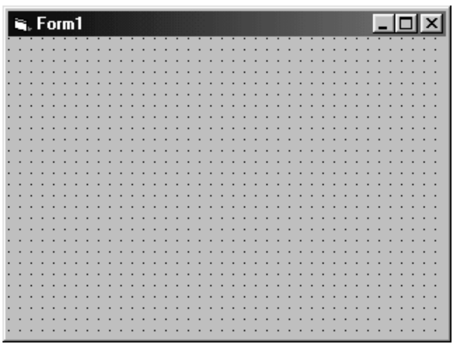
כברירת מחדל, כל הרכיבים בפרויקט מופיעים בארגז הכלים כקבוצה אחת גדולה. אולם, אם תשתמש בהרבה פקדים יקשה עליך לטפל בהם. כדי להקל על הבעיה, Visual Basic מציעה אפשרות להוסיף כרטיסיות לארגז הכלים. (כברירת מחדל קיימת כרטיסיה אחת- General). להוספת כרטיסיה, לחץ לחיצה ימנית על ארגז הכלים, בחר Add Tab מתוך הרשימה הנפתחת ותן לכרטיסיה החדשה שם. לאחר מכן ניתן להעביר פקדים מכרטיסיה לכרטיסיה. תרשים נספח 1.8 מציג את ארגז הכלים עם כרטיסיה בשם Grid Controls שהתווספה.



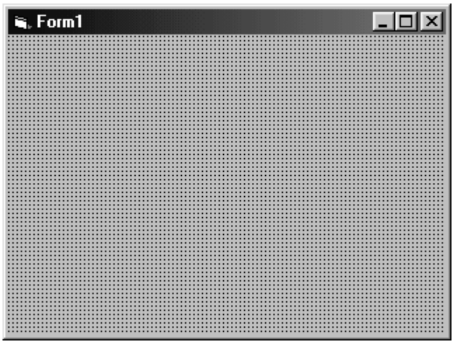
תרשים נספח 1.8: מאפיין נאה של Visual Basic 6 הוא האפשרות לקבץ פקדים בכרטיסיות שונות בארגז הכלים

בד הציור של התוכניות

החלונות שאתה מעצב בתוכניות Visual Basic מכונים טפסים. אפשר לדמיין טופס כבד ציור. הינך משתמש בפריטים מארגז הכלים כדי "לצייר" את ממשק המשתמש על הטופס. הטופס הוא חלק משולחן העבודה והוא השטח העיקרי בו מעצבים את ממשק המשתמש. אם תביט מקרוב על הטופס שבתרשים נספח 1.9 תראה נקודות על הטופס. נקודות אלו יוצרות רשת שמטרתה לעזור לך למקם את הפקדים. כאשר מפעילים את התוכנית, היא בלתי נראית. אפשר לשלוט במרווחים שבין הנקודות על ידי בחירת **Tools**, **Options** ובחירה באפשרות המתאימה בכרטיסיה General של תיבת הדו-שיח. יש אפשרות לא להציג את הנקודות כלל. כברירת מחדל צפיפות הרשת היא 120 טוויפ. אני מעדיף להקטין את הרשת (60X60 טוויפ), דבר המאפשר לי דיוק גדול יותר בהצבת הפקדים על הטופס. רשת בצפיפות זו מוצגת בתרשים נספח 1.10.



תרשים נספח 1.9: בטופס Visual Basic קיימת רשת העוזרת ליישר את הפקדים

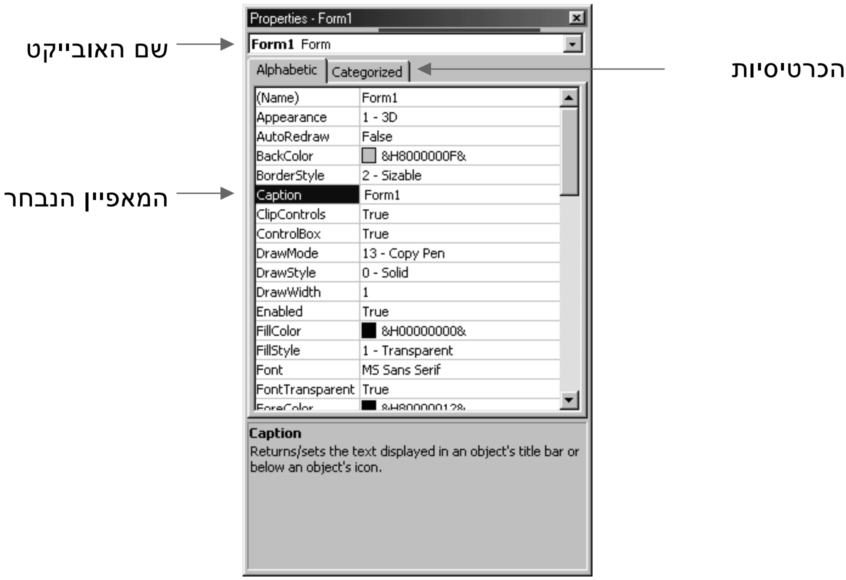


תרשים נספח 1.10: הקטנת הרשת מאפשרת שליטה טובה יותר על מיקום האובייקטים

שליטה על טפסים ופקדים

חלון המאפיינים (Properties window) הוא חלק חשוב ב- Visual Basic. הוא מציג את כל המאפיינים הזמינים לטופס נבחר, פקד או מודול. (ראה תרשים נספח 1.11). אם החלון אינו מוצג, בחר תחילה את האובייקטים שאת מאפייניהם תרצה לראות או לשנות והקש F4. ניתן לראות חלון זה גם על ידי בחירת **View**, **Properties Window**, או בלחיצה ימנית על האובייקט ובחירת **Properties** מהתפריט שיופיע.

המאפיינים קובעים כיצד ייראה או יתנהג פקד התוכנית. חלון המאפיינים מציג את רשימת מאפייני האובייקט הניתנים לשינוי בזמן עיצוב התוכנית, בניגוד למאפייני זמן ההפעלה, הניתנים לשינוי רק בזמן ההפעלה. חלק ניכר מהמאפיינים ניתנים לשינוי בזמן העיצוב וגם בזמן ההפעלה.



תרשים נספח 1.11: חלון המאפיינים מאפשר דרך נוחה לשנות מאפיינים

המאפיין Caption היא דוגמה למאפיין פקד מסוג תווית. ניתן לשנותו בפשטות על ידי הקלדת Hello World בשורת המאפיין Caption שבחלון המאפיינים (שינוי אפשרי רק בזמן העיצוב) או על ידי הוספת המשפט:

```
Form1.Label1.Caption = "Hello World"
```

(שינוי בזמן ההפעלה).

לחלון המאפיינים שתי כרטיסיות. כרטיסיות אלו מאפשרות לקבץ את המאפיינים לפי סדר האלף-בית או בצורה לוגית. שיפור נוסף לחלון המאפיינים הוא הצגת תיאור המאפיין הנבחר בחלון תחתון. מידע זה עוזר להימנע מחיפוש מידע על מאפיינים בקבצי העזרה.

הערה:

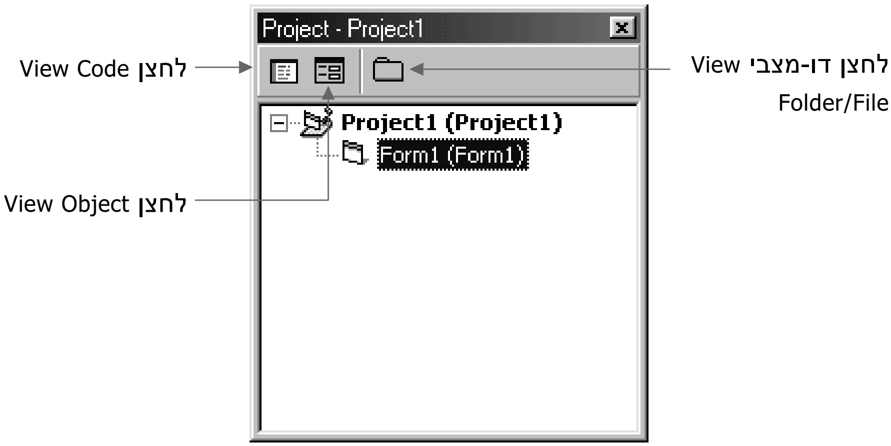
מטעמי נוחות, המאפיין Name של כל אובייקט מופיע בראש הרשימה **ולא** במקומו לפי סדר האלף-בית.

הערה:

לרבים מהפקדים יש שורת ערך (מותאמת) בחלון המאפיינים. לחיצה על הבובה (...) שבשורת הערך פותחת דף מאפיינים מיוחד המכיל את מאפייני זמן עיצוב הפקד, כך שיהיה ניתן לערוך אותו בצורה נוחה.

חלון הפרויקט

חלון נוסף בשולחן העבודה הוא **חלון הפרויקט** (Project Window), כמתואר בתרשים נספח 1.12. חלון זה מציג את רשימת הטפסים, מודולים ורכיבים אחרים בהם משתמשת התוכנית. אם תרצה לראות טופס או מודול, לחץ לחיצה כפולה בזמן העיציב, או לחץ פעם אחת ובחר **View Object** או **View Code**.



תרשים נספח 1.12: חלון הפרויקט מציג את סוגי הקבצים השונים המרכיבים את הפרויקט(ים) הפתוח(ים)

הערה:

אפשר להתייחס לפרויקט כקבוצת שדות מקושרים. הפרויקט מאחד את כל השדות הדרושים ליצירת תוכנית.

בעת שמירת הפרויקט למעשה נשמרת רשימת הקבצים היוצרים את הפרויקט. קובץ הפרויקט עצמו מאוחסן עם סיומת ברירת המחדל VBP (Visual Basic Project). סוגי קבצים אחרים יוצרים את רכיבי הפרויקט. הנפוצים ביותר מופיעים בטבלת נספח 1.3.

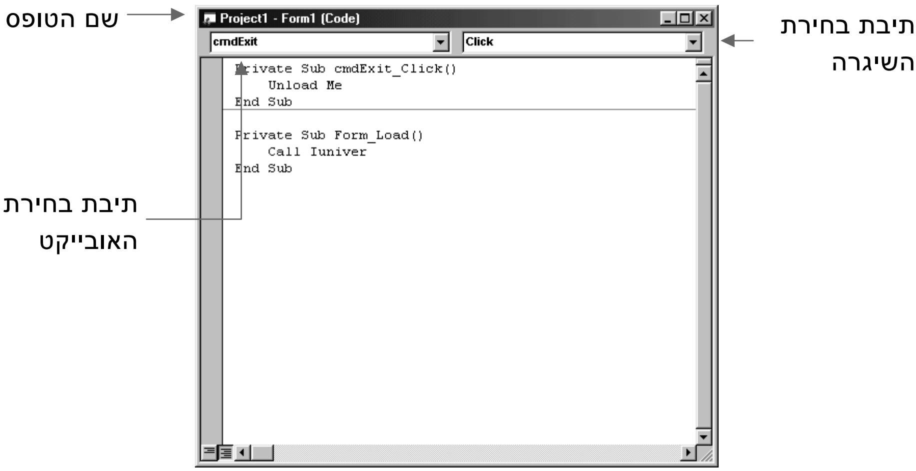
טבלת נספח 1.3: סוגי הקבצים של Visual Basic

סיומת	סוג הקובץ
FRM	טופס Visual Basic
BAS	מודול קוד
CLS	מודול מחלקה
CTL	פקד מתוכנת על ידי המשתמש
DOB	קובץ מסוג ActiveX Document Form

חלון הפרויקט משתמש ברשימה חיצונית להצגת הטפסים והמודולים בפרויקט הפתוח וגם מודולי מחלקה, פקדים מותאמים או עמודי מאפיינים. אפשר לצפות בפרויקט בשני אופנים. תצוגת תיקיה, אליה ניתן לגשת על ידי לחיצה על הלחצן המציג את חלקי הפרויקט מאורגנים לפי נושאים. מצד שני, לחיצה על הלחצן הקיצוני מימין, מציגה את רשימת רכיבי הפרויקט מקבצים בהתאם לשמות הקובץ.

היכן מתבצעת העבודה

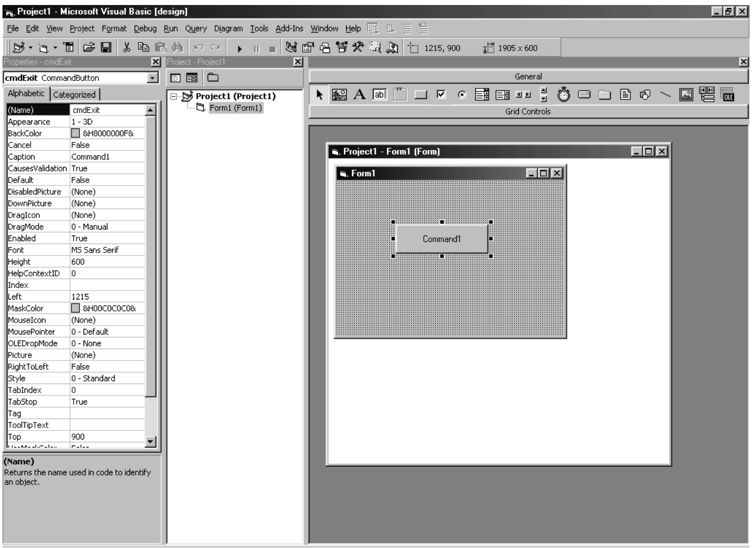
החלק האחרון של שולחן העבודה הוא **חלון הקוד** (Code Window). בחלון הקוד מתבצעת כל עבודת ההקלדה והשינויים לקוד התוכנית, המאפשר לתוכניות לבצע משימות (ראה תרשים נספח 1.13). לכל טופס יש חלון קוד המקושר אליו. פרויקט יכול לשלב גם כמה סוגי קוד עצמאיים המוכרים בשם **מודולים** (Modules). כדי לגשת לחלון קוד, לחץ לחיצה כפולה על הטופס, על אחד האובייקטים שלו, או על הלחצן View Code שבחלון הפרויקט כאשר האובייקט המתאים מסומן.



תרשים נספח 1.13: חלון הקוד הוא המקום בו מקלידים ועורכים את ההוראות שתבצענה את התוכנית

התאמת הסביבה

כפי שלמדת, סביבת הפיתוח של Visual Basic ניתנת להתאמה במידה רבה. רוב החלונות וסרגלי הכלים ניתנים למיקום בקצות החלון הראשי, או יכולים לצוף בכל מקום על המסך. אפשר גם לשנות את גודל החלונות. בפעם הבאה שתפתח את Visual Basic הסביבה תישאר כפי שהשארת אותה. תרשים נספח 1.14 מציג את אחת האפשרויות לארגן את סביבת העבודה.



תרשים נספח 1.14: הפריטים השונים של סביבת העבודה ניתנים לארגון במגוון אפשרויות

נספח 2

אריזת היישום

מה בפרק?

- הידור התוכנית
- אריזת פרויקט Standard EXE
- אריזת רכיבי ActiveX

לאחר שתסיים לכתוב את התוכנית שלך, עליך להפיץ אותה מחוץ לסביבת הפיתוח של Visual Basic כדי לאפשר לאחרים להשתמש בה. הצעד הראשון בתהליך הוא **הידור** (Compilation) של קוד המקור שלך. פעולה זו מיועדת ליצור קובץ EXE (או קובץ DLL/OCX בהתאם לסוג הפרויקט) שניתן יהיה להפיץ למחשבים אחרים. לאחר שתהדר את התוכנית, תוכל ליצור את קבצי ההתקנה של התוכנית, על ידי שימוש בכלי Package and Deployment Wizard (אשף האריזה וההפצה). אשף זה נועד לאפשר לך לארוז את התוכניות ואת קבצי העזר שלהן, באופן שיאפשר להן לפעול על מחשבים אשר Visual Basic אינה מותקנת עליהם. אם התוכנית שלך היא מסוג Standard EXE האשף יצרף אליה תוכנית התקנה. ברוב המקרים משתמש פשוט יוכל להכניס את התקליטור או הדיסקט של התוכנה לכונן המתאים ולהפעיל את תוכנית SETUP.EXE. בנוסף, ליצירת קבצי התקנה של פרויקטים מסוג Standard EXE האשף יכול לשמש גם לטיפול בפקדי ActiveX וברכיבי ActiveX מסוגים אחרים. בנספח זה נתייחס להידור התוכנית, ואחר נציג שימוש באשף לצורך יצירת יישום.

הערה:

Package and Deployment Wizard החליף את Setup Wizard שנכלל בגרסאות קודמות של Visual Basic.

הידור התוכנית

הידור התוכנית הוא הצעד הראשון שיש לבצע בתהליך הפצת התוכנית. כל שצריך לעשות כדי להדר תוכנית הוא לבחור את הפקודה **Make** מתפריט **File**. בחירה בפריט זה תציג את שם הפרויקט ואת הסיומת שתתאים לסוג התוכנית שברצונך ליצור. בעת עבודה עם פרויקטים מסוג Standard EXE או ActiveX EXE תוצג הסיומת EXE. בעת יצירת ActiveX DLL תוצג הסיומת DLL, ובעת יצירת פקד ActiveX תוצג הסיומת OCX. לאחר שתבחר את סוג הפרויקט שאותו תרצה ליצור, תוצג תיבת הדו-שיח Make Project. תיבת דו-שיח זו תאפשר להגדיר את שם ומיקום קובץ היעד. Visual Basic תבצע את שאר העבודה עבורך.

אם תאותרנה שגיאות שתימנענה הידור מוצלח, יש לתקן אותן ולבצע שוב את פעולת ההידור. בכל מצב בו תבצע שינויים בקוד תוכנית שכבר הפצת, יש להפיץ שוב את התוכנית. Visual Basic אומנם תבצע את פעולת ההידור עצמה ללא מעורבות מצידך, אך יש להגדיר מספר אפשרויות בתיבת הדו-שיח Project Properties. את הכרטיסיות Compile ו-Make של תיבת הדו-שיח ניתן להציג גם על ידי לחיצה על לחצן Options שבתיבת הדו-שיח Make Project.

אופטימיזציה של הקוד

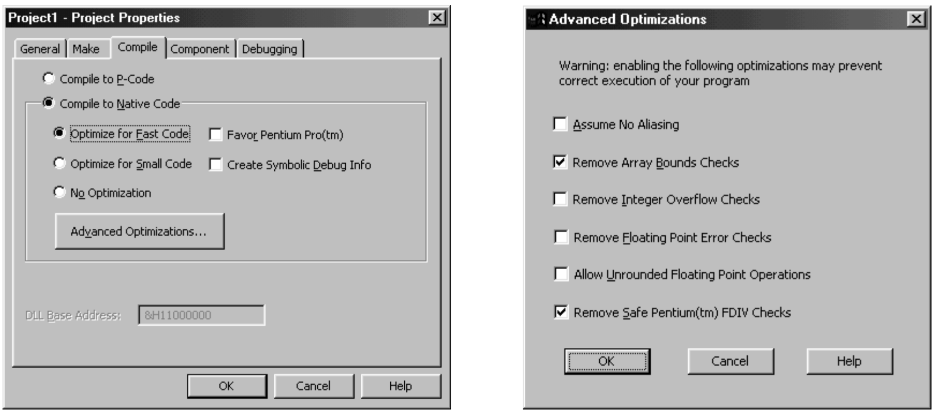
תחילה יש לבחור האם לחולל P-Code או Native Code. P-Code הוא סוג הקוד שאליו הודרו תוכניות שנכתבו ב- Visual Basic מאז גירסה 1, ואילו Native Code הוא אפשרות חדשה שנכללה לראשונה במהדר Visual Basic 5.0. Native Code מותאם בצורה אופטימלית לסוג מעבד מסוים ופועל מהר יותר מאשר P-Code. מאידך, שימוש ב-Native Code מוביל ליצירת קבצי ביצוע גדולים יותר מאלה שנוצרים בשימוש בסוג P-Code. אם תבחר להדר את התוכנית ל- Native Code יש להגדיר גם הגדרות פעולות אופטימיזציה שתבוצענה על ידי המהדר. תוכל להורות למהדר ליצור קובץ ביצוע קטן ככל האפשר או מהיר ככל האפשר, ולחילופין, תוכל גם להורות שלא לבצע פעולות אופטימיזציה כלשהן על הקוד. תוצע גם אפשרות להדר את התוכנית באופן מיוחד לשימוש באמצעות מעבדי Pentium Pro.



טיפ:

כדאי שתבחר להדר Native Code מפני שמהירות היא שיקול חשוב יותר מחיסכון מעט מקום פנוי בדיסק.

לבחירת האפשרויות שתיושמנה על ידי המהדר יש להציג את הכרטיסיה Compile שבתיבת הדו-שיח Project Properties (ראה תרשים נספח 2.1). כדי לסחוט מ- Visual Basic כל טיפת מהירות אפשרית כדאי גם שתציג את תיבת הדו-שיח Advanced Optimizations המוצגת גם היא בתרשים זה. בתיבת דו-שיח זו אזהרה שהשימוש באפשרויות אלו הוא על-אחריותו הבלעדית של המשתמש. למרות האזהרה אני נוהג לסמן את האפשרות Remove Array Bounds Check מפני שקוד התוכנית אמור לבצע בדיקות כאלו, ואת האפשרות Remove Safe Pentium™ Safe FDIV Checks - שמבטלת את ביצוע פעולות התוכנה הדרושות לתיקון ה"באג" החישובי שאותר במעבדי הפנטיום הראשונים. אם בטיחות פעולת התוכנית היא שיקול חשוב עבורך, לא כדאי שתבחר באפשרויות שבתיבת דו-שיח זו.



תרשים נספח 2.1: היעזר בתיבות הדו-שיח Project Properties ו-Advanced Optimizations לביצוע פעולות אופטימיזציה על הקוד שיופק על ידי המהדר

הגדרת שם, כותרת וסמל הפרויקט

פעולה חשובה בתהליך הידור הפרויקט היא לוודא שהמאפיינים Name ו-Title של הפרויקט מכילים ערכים בעלי משמעות. כברירת מחדל, Visual Basic תקצה עבור הפרויקט שלך שם גנרי כגון Project1. גם אם תשנה את שם קובץ היעד של הפרויקט באמצעות תיבת הדו-שיח Make Project, עדיין יש לשנות את שם הפרויקט ואת הכותרת שלו בתוך הפרויקט. אם כבר התנסית בכתיבת רכיבי ActiveX כדוגמת קבצי ActiveX DLL המכילים מחלקות ציבוריות, אתה בוודאי מודע לחשיבות שם הפרויקט, ובוודאי גם התנסית בהצבת ערך בעל משמעות במאפיין זה. כותרת הפרויקט חשובה מפני שהיא עשויה להופיע ב- Windows Task List או בהודעות שגיאה. תיבות הטקסט המשמשות להגדרת המאפיינים Name ו-Title נמצאות בכרטיסיות General ו-Make שבתיבת הדו-שיח Project Properties, בהתאמה.

הערה:

ככלל, אין סיבה להציב ערכים שונים במאפיינים Name ו-Title של הפרויקט. אם מאפיין Title לא הוגדר, Visual Basic תציב בו את שם הפרויקט.

בטרם תפיץ את היישום, כדאי שתגדיר **סמל** (Icon) עבורו, בעזרת כרטיסיית Make שבתיבת הדו-שיח Project Properties. סמל אומנם אינו רכיב הכרחי ביישום, אך השימוש בסמל יעניק ליישום שלך חזות מקצועית יותר. כל מה שיש לעשות כדי להקצות סמל לטופס מסוים הוא לבחור את שם הטופס בתיבה הנפתחת Icon.

הכנות ליצירת תוכנית התקנה

לאחר שתהדר את היישום בהצלחה, נסה את הגירסה המהודרת על המחשב, כדי לוודא שהיא אכן תפעל מחוץ לסביבת הפיתוח המשולבת של Visual Basic. יישום Standard EXE הרץ מתוך סייר Windows. אם תיצור רכיב ActiveX נסה את הרכיב בפרויקט Visual Basic או דף Web. לפני שתפיץ את היישום, יש לבצע עליו בדיקות יסודיות יותר, תוך שימוש במחשב שונה מזה שעליו הוא פותח.

אני גם ממליץ להדר את הפרויקט בתיקיה שונה מזו שבה פיתחת אותו. ביצוע פעולה כזו עשוי לסייע לך באיתור קבצים נוספים (כגון קבצי INI או קבצי תמונה), שקוד היישום פונה אליהם. לאחר שתזהה את הקבצים האלה ותיצור עותק פועל של היישום שהודר, תוכל לבצע את הצעד הבא - שימוש ב- Package and Deployment Wizard.

אריזת פרויקט Standard EXE

הידרת את הקוד לקובץ הפעלה (Executable). אך הוא אינו יכול לפעול לבד. משתמשי היישום יידרשו להיעזר ב**קובץ זמן ריצה** (Runtime File) של Visual Basic (לדוגמה, קובץ DLL) שהותקן כיאות. כדי להשתמש ביישום בגירסה Visual Basic 3, היה על המשתמש להעתיק את קובץ VBRUN300.DLL, ואת קובץ הפעלת (EXE) היישום. כיום יש להעתיק את קובץ DLL ולרשום אותו ברישום (Registry) Windows.

מאידך, איכות תוכנית העזר ליצירת קבצי התקנה ב- Visual Basic השתפרה מגירסה לגירסה, והגירסה החדשה המכונה Package and Deployment Wizard היא הגירסה הטובה ביותר עד כה. התוכנית כוללת ממשק משתמש חדש ותמיכה באפשרויות נוספות להתאמה אישית של יישומים, שלא נתמכו על ידי גרסאות קודמות. מטרת התוכנית היא "אריזת" התוכנית שלך ביחד עם קבצי העזר שלה, באופן שיאפשר להתקין אותה מדיסק או מה-Web, באותו אופן בו מתקינים תוכנות מדף.

יצירת האריזה של יישום Standard EXE

סביר להניח שבהתקנת Visual Basic נכלל גם קיצור דרך ל- Package and Deployment Wizard. מסך הפתיחה של האשף, הנראה בתרשים נספח 2.2 הוא נקודת ההתחלה ליצירת תוכניות התקנה לכל סוגי היישומים שניתן לכתוב ב- Visual Basic. סעיף זה יתמקד בצעדים הדרושים ליצירת תוכנית התקנה לפרויקט Standard EXE. כדי להתייחס לסעיף כדוגמה במבנה צעד-אחר-צעד, יש ליצור תחילה פרויקט פשוט מסוג Standard EXE, לשמור ולהדר אותו. אחר תוכל להפעיל את האשף.



תרשים נספח 2.2: האפשרות החשובה ביותר במסך הפתיחה של Package and Deployment היא הלחצן Package המשמש לבניית חבילות התקנה ליישומים

בחירת קובץ פרויקט

הפעולה הראשונה שיש לבצע באשף היא בחירת קובץ פרויקט (קובץ VBP) בעזרת לחצן Browse. אחר לחץ על לחצן Package כדי להתחיל ביצירת החבילה.

הערה:

אם טרם הידרת את הפרויקט שעבורו אתה בונה את החבילה, תוצג תיבת הודעה אשר תשאל האם ברצונך שהאשף יהדר אותו עבורך. לא תוכל להמשיך בפעולת בניית החבילה כל עוד לא תהדר את הקובץ. אני ממליץ להדר את הפרויקט בסביבת הפיתוח של Visual Basic ולא באשף, מפני שהידור הפרויקט באשף מוסיף צעד נוסף.

בחירת סוג חבילה

לאחר שהאשף יקרא את הנתונים המאוחסנים בקובץ VBP שבחרת, הוא יציג את מסך Package Type שמוצג בתרשים נספח 2.3. מסך זה יאפשר לבחור את Package Type (סוג החבילה) שתיצור. זכור שבמקרה זה המושג חבילה הוא מילה נרדפת לאוסף הקבצים המשמשים להתקנת היישום שלך.

הערה:

אם **יוצג** מסך Packaging Script במקום מסך Package Type בחר None **ו**לחץ על לחצן Next. מסך זה **יוצג** אם שמרת קודם לכן הוראות אריזה כלשהן.



תרשים נספח 2.3: סוגי האריזות שיוצעו בעת אריזת פרויקט מסוג Standard EXE הם Standard Setup Package ו-Dependency File

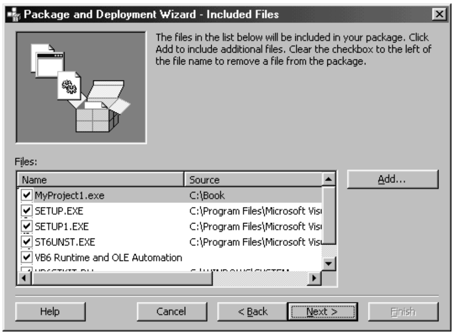
אם תבחר את הסוג Dependency File, האשף לא ייצור תוכנית התקנה ניתנת להפעלה (EXE). Dependency File (קובץ תלות) מכיל את הנתונים על הקבצים אשר מאפשרים לשלב גרסת התקנה של היישום בחבילת התקנה של פרויקט אחר. בחר את האפשרות Standard Setup Package, אלא אם כן אתה יוצר תוכנית התקנה אחת שתשמש להתקנת מספר פרויקטים. לצורך הדוגמה בחר אפשרות זו ולחץ על Next.

הגדר תיקיית חבילה

הנתון הבא שיש להזין לאשף הוא **תיקיית חבילה** (Package Folder). תיקיית חבילה היא התיקיה שבה האשף ייצור את קבצי ההתקנה שלך. אם אינך מעוניין שהקבצים יאוחסנו בתיקיה שקיימת בדיסק, תוכל להזין שם תיקיה חדשה. כברירת מחדל תיווצר תיקיה חדשה בשם Package בתיקיה בה מאוחסן קובץ הפרויקט.

בחר את הקבצים

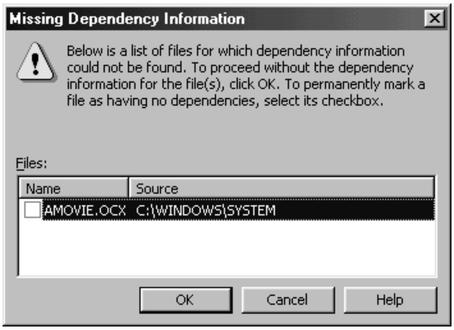
לאחר שתבחר את התיקיה, יוצג המסך Included Files שמוצג בתרשים נספח 2.4. המסך יפרט את הקבצים שייכללו בחבילת ההתקנה ויאפשר להוסיף קבצים נוספים. אם יש קבצים שברצונך להפיץ עם התוכנית, למשל קבצי INI או BMP, תוכל להוסיפם לרשימה בעזרת לחצן Add. בזמן העבודה במסך זה יוצגו תיאורי כלים שיציינו האם קובץ מסוים הוסף לחבילה באופן ידני, או האם הוא נדרש על ידי קובץ אחר.



תרשים נספח 2.4: תיבת הסימון מציינת שהקובץ (והקבצים התלויים בו, אם ישנם כאלה) ייכללו בחבילת ההתקנה של היישום

נניח שתרצה להוסיף לפרויקט קובץ EXE שנוצר באמצעות Visual Basic, קובץ DLL, או קובץ OCX, למשל אם חילקת את היישום למספר פרויקטים, במקרה וכללת בו "חבילה מנהלתית" לניהול כניסת משתמשים למערכת. אם לא יצרת חבילת התקנה עבור כל אחד מהפרויקטים האחרים, ייתכן שתוצג תיבת הדו-שיח Missing Dependency Information, שמוצגת בתרשים נספח 2.5.

הודעת אזהרה זו הוצגה כדי לאפשר לך לוודא שתוכנית ההתקנה תפעל. לדוגמה, אם החלטת לשלב ביישום קבצי EXE נוספים, יש צורך גם בקבצי DLL ו-OCX הדרושים לאותם קבצים. אם תוצג הודעה כזו, תוכל לבחור בין יצירת נתוני תלות בעזרת האשף, לבין התעלמות ממנה אם תדע בוודאות שלא יידרשו קבצי עזר כלשהם. אם תסמן תיבות סימון כלשהן בתיבת הודעה זו (פעולה שאינני ממליץ לעשות), האשף לא יציג שאלות נוספות בקשר לנתוני תלות.



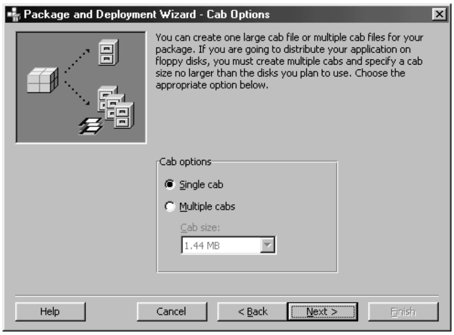
תרשים נספח 2.5: תיבת האזהרה המוצגת כאן מציינת שהאשף לא איתר קובץ קשרי תלות (Dependency File - DEP) עבור הפריט שנבחר

בחר אופציות CAB

הצעד הבא הוא בחירת סוג קבצי CAB שברצונך ליצור. מסך Cab Options שמוצג בתרשים נספח 2.6 מאפשר ליצור קבצי CAB בהתאם לסוג המדיה שבה תשתמש להפצת היישום. להתקנה מכונן רשת או מתקליטור, בחר Single Cab, להתקנה מדיסקטים בחר Multiple Cabs.

הערה:

בגרסאות קודמות של Visual Basic קבצי ההתקנה נדחסו ושמותיהם שונו לשמות שהכילו קווים תחתיים (Underscore) כדי לחסוך מקום. הגירסה העדכנית של Visual Basic נעזרת בפורמט CAB. קבצי CAB (הדומים לקבצי ZIP) מכילים גרסאות דחוסות של קבצי ההתקנה. ניתן לבצע פעולות על הקבצים הדחוסים משורת הפקודה של DOS על ידי שימוש בתוכנית השירות Extract או בסביבת Windows על ידי שימוש בתוכנית השירות CabView (ב- Windows 98 אין צורך ב-CabView).



תרשים נספח 2.6: דחס את קבצי ההתקנה לקובץ CAB אחד או לכמה קבצים קטנים

בזאת סיימת להגדיר את הנתונים החיוניים ליצירת תוכנית התקנה לפרויקט מסוג Standard EXE. המסכים הנותרים, שלא היו זמינים בגרסאות קודמות, יאפשרו לך לבצע פעולות נוספות להתאמה אישית של תוכנית ההתקנה.

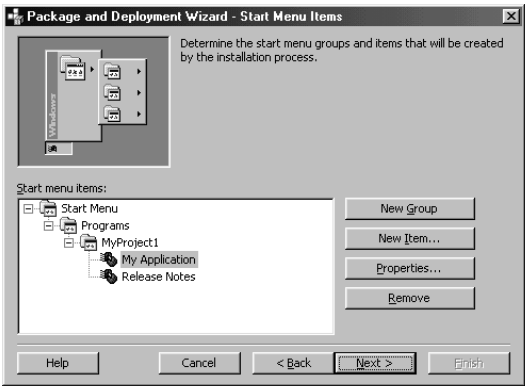
הגדרת כותרת התקנה

לחץ על לחצן Next של מסך Cab Options כדי להציג את מסך Installation Title (כותרת ההתקנה). מסך זה, שאינו מוצג בתרשים, יאפשר להגדיר כותרת עבור תוכנית ההתקנה כגון Executive Reporting System, הכותרת שתגדיר, תוצג ברקע במהלך פעולת תוכנית ההתקנה.

יצירת קיצורי דרך וקבוצות יישומים

מסך Start Menu Items (ראה תרשים נספח 2.7) הוא כלי התקנה חדש נוסף, שנכלל לראשונה ב- Visual Basic 6. המסך מאפשר לשלוט על סמלי קיצור דרך ועל קבוצות יישומים שתוכנית ההתקנה תיצור. בגרסאות הקודמות, האשף יצר רק קיצור דרך אחד לקובץ ההפעלה הראשי של הפרויקט.

היעזר בלחצנים New Group ו- New Item כדי ליצור קבוצות יישומים וקיצורי דרך להתקנה. כל אחד מהלחצנים יציג תיבת דו-שיח שתאפשר להזין תיאור ונתונים נוספים. לדוגמה, תוכל להשתמש במסך זה כדי להגדיר קיצור דרך לקובץ הערות התקנה שתספק עם היישום, או כדי להגדיר כתובת URL שתעביר לאתר מסוים ב-Web. במקרה כזה לחץ על הלחצן New Item, כדי לבחור את קובץ היעד מרשימת הקבצים שבחבילה, ולהגדיר כותרת לסמל קיצור הדרך החדש שיווצר.



תרשים נספח 2.7: תוכנית ההתקנה יכולה ליצור עבורך קיצורי דרך וקבוצות יישומים

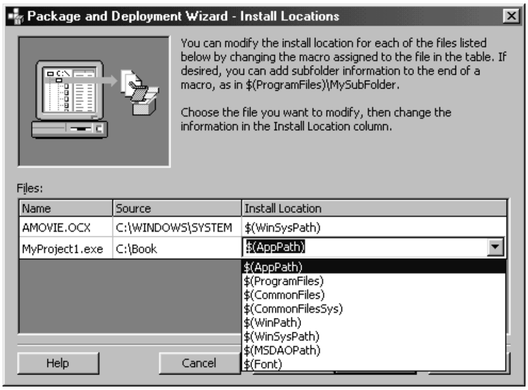
הערה:

לחצן Properties יבצע פעולות שונות בעת יצירת Group ו-Item. ביצירת Item תוכל לערוך את הקובץ, כותרת הפריט ואת התיקיה הפעילה. בעת יצירת קבוצות יישומים בסביבת Windows NT תוכל להגדיר האם הקבוצה תהיה Common (קבוצה משותפת - שתוצג בתפריט Start של כל חשבונות המשתמש) או Private (קבוצה פרטית - שתוצג רק בתפריט Start של חשבון המשתמש הפעיל).

הגדר מיקומי התקנות

לאחר שתסיים להגדיר פריטי תפריט התחלה, לחץ על לחצן Next ויוצג מסך נוסף, Install Locations, שבו תוכל לבצע פעולות התאמה אישית. במסך זה, שמוצג בתרשים נספח 2.8 קבע את הספריות שבהן יותקנו קבצי היישום.

סוגי קבצים מסוימים, כגון קבצי DLL ו-OCX מותקנים לרוב בתיקיה בה מותקנת מערכת ההפעלה Windows. קובץ ההפעלה של היישום מועתק באופן דומה לתיקיה שהמשתמש בוחר בעת התקנת היישום. משום שתיקיות אלו משתנות ממחשב למחשב, האשף נעזר במאקרוס לצורך ייצוגן.



תרשים נספח 2.8: תוכל להגדיר בעצמך את המקום בו יותקן כל קובץ

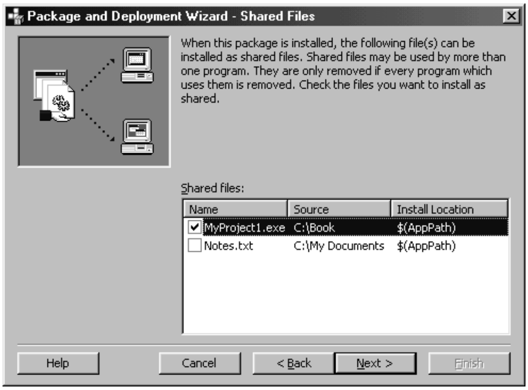
תוכל להתייחס למאקרוס, הניתנים לזיהוי לפי התחביר $(Macroname) כאל סמלים שתוכנית ההתקנה תציב במקום שמות ספריות הקיימות באותו מחשב. לדוגמה נתייחס לתיקיה System של Windows, שנתיב ברירת המחדל שלה ב- Windows 95 הוא C:\Windows\System, לעומת Windows NT שם הוא C:\WINNT\SYSTEM32. הגדרת נתיב מותאם בעת התקנת Windows גורמת לתיקיה System לקבל את הנתיב D:\MyWin95\System או נתיב דומה. אך המאקרו $(WinSysPath) יאתר את התיקיה System בכל מחשב.

סמן רכיבים משותפים

תיבת הדו-שיח הבאה Shared Files, תאפשר לסמן קבצים מסוימים כ**רכיבים משותפים** (Shared Components). רכיבים משותפים הם לרוב קבצי DLL או קבצים מסוגים אחרים אשר משמשים כמה תוכניות במקביל. לדוגמה, אם התקנת 10 יישומי Visual Basic על מחשב מסוים, כל היישומים "חולקים ביניהם" את קובץ זמן הריצה MSVBVM60.DLL. קובץ המסומן כקובץ משותף אינו מוסר בעת הסרת התוכנית, אלא אם כן תוכנית ההסרה מזהה שלא קיימים יישומים אחרים אשר משתמשים באותו קובץ. Visual Basic היא שמסמנת את הקבצים המשותפים, ואילו הרישום (Registry) מנהל את המעקב אחר היישומים שמשתמשים באותם קבצים משותפים. ברוב המקרים יהיה כדאי לבחור את הגדרות ברירת המחדל (ראה תרשים נספח 2.9).

הערה:

הגדרת קובץ הפעלה כקובץ משותף, למרות שאינו מיועד להיות כזה, תימנע את הצגת שתי תיבות אזהרה שתתייחסנה למקרי החלפת קובץ קיים. תיבות האזהרה תצגנה הודעות סותרות (כגון ?Cancel Setup ו-?Continue Setup) והן עלולות לבלבל משתמשים מסוימים. בכך שתגדיר את הקובץ כרכיב משותף, תורה למעשה לתוכנית ההתקנה שהיא אינה זקוקה לאישור מהמשתמש כדי לדרוס את הקובץ.



תרשים נספח 2.9: אין הכרח להסיר קבצים משותפים בעת הסרה יישום

הערה:

תוכל להסיר יישומים על ידי בחירה באפשרות **הוספה/הסרה של תוכניות** מ**לוח הבקרה** של Windows.

שמור את תסריט האריזה

לאחר שתסיים להגדיר את הנתונים הדרושים לאשף, יוצג מסך Finish (סיום). אם שינית רבות מההגדרות הסטנדרטיות, תוכל לקרוא ל- Packaging Script (תסריט האריזה) בשם בעל משמעות לשימוש עתידי. אם לא תרצה לשנות שם קובץ ה-Script, תוכל פשוט ללחוץ על לחצן Finish וליצור את קבצי ההתקנה של היישום.

טיפ:

לאחר שיצרת תסריט אריזה, בפעם הבאה שתארוז יישום, **יוצג** מסך Packaging Script של האשף. תוכל לבחור לטעון תסריט אריזה קיים, או להתחיל ביצירת Script חדש על ידי בחירה באפשרות None. אם תרצה למחוק Script קיים או לשנות את שמו, תוכל לבחור באפשרות Manage Script במסך הראשון.

לאחר שקבצי ההתקנה ייווצרו, יוצג Packaging Report (דוח אריזה) כמו זה הנראה בתרשים נספח 2.10.



תרשים נספח 2.10: Packaging Report יאפשר לדעת היכן אוחסנו קבצי ההתקנה

סיים את תהליך האריזה

כאשר תסגור את חלון Packaging Report תחזור למסך הראשי של האשף. בזאת סיימת את תהליך האריזה, וכעת תוכל לנסות את תוכנית ההתקנה שיצרת על ידי הפעלת הקובץ SETUP.EXE מהתיקיה שבה בנית את הפרויקט.

אם יש ליצור תוכנית התקנה שתשמש משתמשים רבים, כדאי שתבצע בדיקות מקיפות על היישום, כדי לוודא שחבילת ההתקנה מכילה את כל הרכיבים הדרושים. לא תוכל להסתפק בבדיקת היישום על המחשב שלך, מפני שייתכן שקבצי DLL ו-OCX הדרושים לשימוש ביישום היו מותקנים על המחשב הזה עוד קודם לכן.

טיפ:

אחת הדרכים שבהן תוכל לוודא שחבילת ההתקנה של היישום אכן מכילה את כל הקבצים הדרושים לשם שימוש בו, היא על ידי בדיקת החבילה על מחשב שעליו מותקנת רק מערכת הפעלה. אך זו שיטה הכרוכה בבעיות מסוימות מפני שכל בדיקת התקנה תבצע שינויים באותה מערכת. אני מציע שתרכוש תוכנה שמאפשרת לשחזר את המצב ההתחלתי של המחשב על ידי שימוש בקובץ Image File (קובץ תמונה), אשר תאפשר לך לבצע בדיקות התקנה תוך שימוש במיגוון תצורות תוכנה.

סקירה מפורטת של תהליך ההתקנה

זה עתה סקרנו את התהליך המשמש ליצירת מערכת קבצי התקנה המכונה גם **חבילה** (Package). ניתן להניח שרוב המשתמשים התנסו כבר בהתקנת תוכנות. מי לא התנסה עדיין בישיבה מול מחשב שהציג מד התקדמות זוחל במהלך התקנה? רוב המשתמשים מתייחסים לתוכניות התקנה כלתוכניות המבצעות פעולה פשוטה. אך על מפתח תוכנת התקנה להבין שהתוכנה אינה מעתיקה קבצים בלבד. בסעיפים הבאים נסקור את סוגי הקבצים שנוצרים בשימוש ב- Package and Deployment Wizard ואת הפעולות המתרחשות בעת תהליך ההתקנה.

קבצי התקנה

הפעולות המשמשות ליצירת חבילת התקנה של יישום מסוג Standard EXE, שתוארו בסעיפים בקודמים, יוצרות שתי תיקיות חדשות בדיסק:

- **Package** (תיקיית חבילת ההתקנה) המכילה את קבצי ההתקנה הדחוסים בפורמט CAB, את קובץ ההפעלה של תוכנית ההתקנה, את Dependency File (קובץ קשרי התלות) ואת קובץ בקרת ההתקנה SETUP.LST. עליך להעתיק (או להפיץ) את תוכן תיקיית Package למחשב שבו ברצונך להתקין את התוכנה.
- **Support** (תיקיית קבצי עזר) מכילה את קבצי ההתקנה כאשר הם פרוסים (לא דחוסים). אין צורך להפיץ תיקיה זו בעת הפצת היישום.

הערה:

בעת יצירת קבצי התקנת רכיב ActiveX ייוצרו קבצי התקנה נוספים. התקנת רכיבי ActiveX תתואר בהמשך.

SETUP.LST הוא אחד הקבצים שאשף ההתקנה יצר. קובץ זה הוא קובץ הבקרה של תהליך התקנת פרויקט מסוג Standard EXE. ניתן להציג את תוכנו באמצעות עורך טקסט, כמתואר בתרשים נספח 2.11.

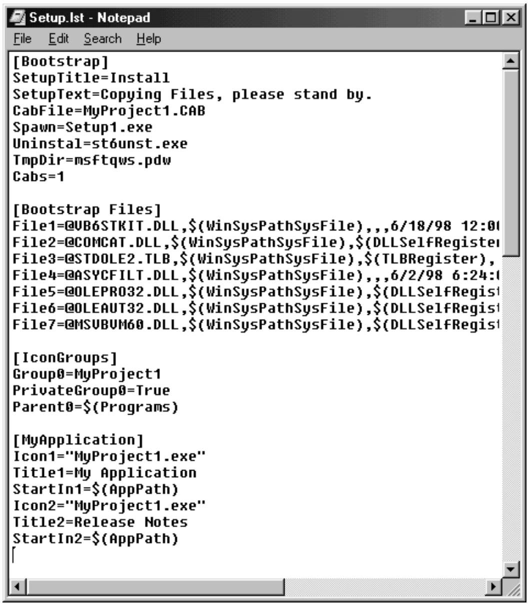
קובץ SETUP.LST

אל תיבהל מתוכנו הבלתי ברור של הקובץ. למעשה, המבנה הוא ברור מאוד. אביא כעת סקירה מפורטת על אופן השימוש בקובץ:

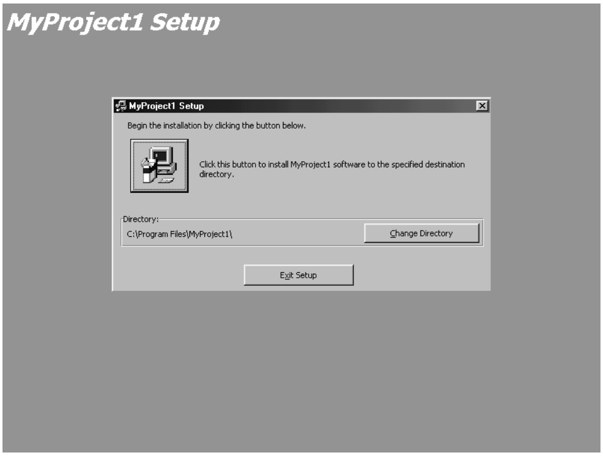
1. כאשר קובץ SETUP.EXE מופעל על ידי המשתמש, הקבצים ששמותיהם מאוזכרים בקטע [BootStrap Files] של קובץ זה, מועתקים למחשב היעד, נפרסים ונרשמים ברישום. במקרים מסוימים, אם קבצים מסוימים אינם מעודכנים, המשתמש עשוי להידרש לאתחל את המחשב. משום שייתכן שקבצי זמן ריצה של Visual Basic לא יהיו קיימים, יש לכתוב את תוכנית ההתקנה בשפה שתאפשר לה לפעול גם בלעדיהם.

   הקבצים המעטים האלה הם המינימום הנדרש להפעלת תוכנית Visual Basic, במיוחד כאשר מדובר בתוכנית SETUP.EXE שמבצעת את רוב העבודה הקשורה בהתקנת היישום. זה הייעוד העיקרי של SETUP.EXE, שמתפקדת כ"מעטפת" עבור התוכנית SETUP1.EXE. SETUP1.EXE מסופקת עם Visual Basic והאשף כולל אותה בקבצי CAB באופן אוטומטי.
2. התוכנית SETUP1.EXE תציג הודעת ברכה ותבקש מהמשתמש להגדיר תיקיה שבה יותקן היישום, כמתואר בתרשים נספח 2.12.
3. אם המשתמש יבחר להמשיך, התוכנית תיצור את התיקיה בה יותקן היישום ותתחיל בהעתקת הקבצים. הקבצים שיועתקו מפורטים בקטע [Setup1 Files] של קובץ SETUP.LST. עבור כל קובץ בקטע זה מפורטים פרמטרים רבים. פרמטרים

אלה מספקים לתוכנית ההתקנה נתונים על כל קובץ. בין היתר, הנתונים כוללים את שם התיקיה בה יותקן הקובץ ואת מספר הגירסה שלו.



תרשים נספח 2.11: האפשרויות שבחרת באשף מוצגות בקובץ SETUP.LST



תרשים נספח 2.12: המשתמש יוכל לבחור תיקיה שבה יותקן היישום

הערה:

אם לא תרצה לאפשר למשתמש לבחור בעצמו תיקיית יעד, הוסף את השורה ForceUseDefDir=1 לקטע [Setup] של קובץ SETUP.LST. אם תרצה להגדיר תיקיית ברירת מחדל אחרת, תוכל לשנות את שורת DefaultDir.

אזהרה:

מספר הגירסה של התוכנית חשוב מאוד. אם לא שינית את מספר הגירסה העדכני בתיבת הדו-שיח Project Properties, תוכנית ההתקנה תניח שבמחשב המשתמש מותקנת הגירסה הרצויה, כך שהיא לא תדרוס אותה. אם תיתקל בבעיות תוכל לעיין בקובץ Installation Log (קובץ יומן התקנה), כדי לוודא אילו קבצים אכן הועתקו.

4. כשתוכנית SETUP1.EXE תסיים להעתיק את הקבצים, היא תנסה לרשום את חלקם ברישום. הבדיקה האם יש לרשום קובץ תתבסס על אחד מהפרמטרים שבקובץ SETUP.LST, ברוב המקרים, הפרמטר $(DLLSelfRegister) או הפרמטר $(EXESelfRegister).

הערה:

אם יש לרשום קובץ מסוים באופן ידני, תוכל להיעזר לשם כך בתוכנית השירות REGSVR32 שמסופקת כחלק מ-Windows. ניתן לרשום קבצים מסוג ActiveX EXE על ידי הפעלתם בשורת הפקודה, תוך הכללת הפרמטר /REGSERVER בשורת ההפעלה.

5. SETUP1.EXE תיצור סמלי קיצור-דרך עבור התוכנית. שמות קבוצות היישומים מפורטים בקטע [Icon Groups] של קובץ SETUP.LST.

6. תוכנית התקנה תבנה **קובץ יומן התקנה** (Installation Log File) שבו תפורטנה הפעולות העיקריות בתהליך ההתקנה (העתקה, רישום קבצים וכך הלאה), שיישמר בתיקיה בה יותקן היישום. קובץ היומן נועד לשימוש תוכנת ההסרה, אך ניתן להשתמש בו גם אם יש צורך לטפל בבעיות שהתעוררו במהלך ההתקנה.

בנייה חוזרת של קובץ CAB

בוודאי שמת לב שקבצי ההתקנה נדחסו לקובץ CAB, או לכמה קבצי CAB. אם תבצע שינוי ולו הקטן ביותר בתוכנית שלך, יש להדר שוב את התוכנית ולאחסן את קובץ ההפעלה המעודכן בקובץ CAB. זוהי הסיבה שבעטיה האשף יוצר עותק נוסף, לא דחוס, של סדרת קבצי ההתקנה של היישום, שאותו הוא מאחסן בתיקיה Support. האשף גם ייצור **קובץ אצווה** (Batch File), שיקבל את השם שהגדרת לפרויקט, למשל PROJECT1.BAT. אם תרצה לעדכן את חבילת ההתקנה מבלי להפעיל שוב את האשף, תוכל להיעזר בקובץ אצווה זה.

לבנייה חוזרת של קובץ CAB, העתק את קובץ EXE החדש לתיקיה Support ולאחר מכן לחץ לחיצה כפולה על קובץ האצווה. בתיקיה Support ייבנה קובץ CAB חדש שאותו יש להעתיק אל תיקיית Package, תוך דריסת קובץ CAB הקיים.

התאמה אישית של ההתקנה

קובץ SETUP.LST אומנם מאפשר לבצע עליו פעולות התאמה מסוימות, אך ייתכן שתוכנית ההתקנה הסטנדרטית לא תתאים לצרכיך. לדוגמה, ייתכן שתרצה ליצור תוכנית "מעטפת" שתעטוף את תוכנית SETUP.EXE, כדי שזו תבצע פעולות אחרות. תוכל למשל להגדיר באופן זמני את שרת ההתקנה ככונן ברשת, להפעיל את SETUP.EXE ולהתנתק מכונן זה בסיום ההתקנה. אם תפיץ תוכנית כזו למשתמשים באמצעות דואר אלקטרוני, הם לא יידרשו לנושא התקשרות לכונן הנכון. אפשרות נוספת היא יצירת מסד נתונים שיכיל נתונים על תאריכי התקנת התוכנה אצל כל אחד מהמשתמשים. גם במקרה כזה יש לכתוב את SETUP.EXE באופן שלא ידרוש שימוש בקבצי זמן הריצה של Visual Basic 6, כפי שעשית בעת כתיבת SETUP.EXE.

אם תרצה להשתמש בתוכנית התקנה שאינה SETUP1.EXE, תוכל עדיין להיעזר בה לצורך התקנת קבצי RUNTIME DLL של Visual Basic 6. החלף את SETUP1.EXE בתיקיה Support ובנה מחדש את קובץ CAB. תוכל גם לגרוע את הקטע [Setup1 Files] מקובץ SETUP.LST אם תיווכח שקטע זה אינו דרוש לתוכנית ההתקנה שלך.

הערה:

Microsoft מספקת קוד מקור של SETUP1.EXE כחלק מ- Visual Basic. אם תרצה לעיין בו, פתח את הקובץ SETUP1.VBP, בנתיב Wizards\PDWizard\Setup1. אני ממליץ שתיצור עותק גיבוי של הקבצים לפני שתבצע בהם שינויים כלשהם.

התקנות מוצלחות

אם ליישום יש משתמשים רבים, התקנתו עשויה להיות פעולה המלווה בבעיות. בעת התקנת תוכניות המיועדות לפעול בסביבת שרת/לקוח סביר להניח שיש גם להתקין מנהלי התקן (Drivers) עבור מסד נתונים ולהגדיר מקורות נתונים (Data Sources), פעולה שתהיה שלב נוסף בתהליך ההתקנה. אפילו התקנות המיועדות להתבצע מהאינטרנט, שאמורות להיות מבוצעות באופן אוטומטי על ידי Internet Explorer עשויות לדרוש טיפול ידני בהגדרות אבטחה.

אני מניח שרבים יסכימו שאין לדרוש ממשתמשים לבצע "שמיניות באוויר" כדי להתקין יישום. בעת יצירת תוכנית התקנה כדאי לשאוף ליצור תוכנית התקנה חד-שלבית שתבצע את כל הפעולות באופן המהיר והיעיל ביותר האפשרי. כדי שתוכל להשיג יעד זה, יש לנקוט לעיתים בפעולות יצירתיות, ולא להסתפק בתוצר הסטנדרטי שהאשף מספק. אך זכור שנקיטת פעולות אלו תעניק ליישום מראה מקצועי יותר.

תוכל לחסוך את הטיפול בבעיות הקשורות בפעולת היישום על ידי ביצוע פעולות לבדיקה וניסוי מקיפים של היישום בטרם הפצתו. כדאי לנסות את היישום על מחשב שניתן להחזיר את מערכת ההפעלה שלו למצבה ההתחלתי על ידי שימוש בתוכנת Ghost. כדאי גם שתנסה את היישום באמצעות מערכות הפעלה שונות, ושתבדוק הבדלים העשויים להתעורר בין התקנות שידרוג לבין התקנות חדשות. כמו כן חשוב מאוד שתבדוק את האינטראקציה בין היישום ובין חבילות תוכנה אחרות.

אריזת רכיבי ActiveX

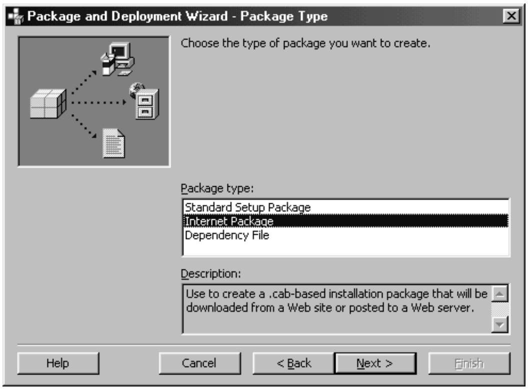
בסעיפים הקודמים למדת על השימוש באשף Package and Deployment לאריזת פרויקט מסוג Standard EXE. התהליך הדרוש לשם אריזת רכיבי ActiveX כגון פקדי ActiveX (קבצי OCX) ו-ActiveX DLL's דומה, אך האשף תומך גם בכמה אפשרויות ייחודיות לאריזת פרויקט מסוג זה.

הורדה מהאינטרנט

בעבודה עם רכיב ActiveX מוצעת לך אפשרות ליצור תוכנית התקנה רגילה שתופעל בעקבות הוראה מפורשת של המשתמש, אך גם תוצע לך אפשרות ליצור חבילת התקנה המיועדת להורדה מהאינטרנט. במקרה שתיצור חבילה כזו, Internet Explorer יהיה הגורם שיטפל בהתקנת הרכיב. אם תרצה ליצור חבילה להתקנה מהאינטרנט, בחר את האפשרות Internet Package ממסך Package Type המוצג בתרשים נספח 2.13.

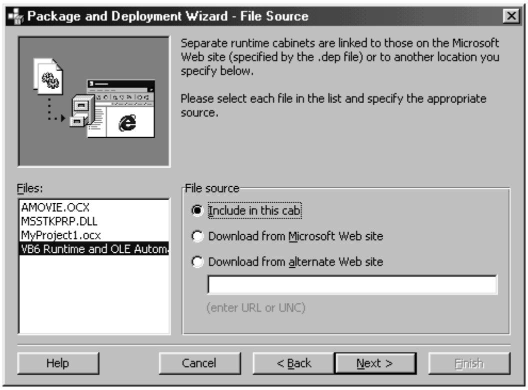
יש להחליט גם אילו קבצים יש לשלב בקובץ CAB.

כדי שמשתמשים יוכלו להתקין את רכיב ActiveX, יידרשו גם קבצי DLL הרלוונטיים של Visual Basic. כברירת מחדל, האשף ייצור חבילה שתורה לדפדפן להוריד קבצים אלה מאחד מאתרי Microsoft, כך שניתן לומר שכאשר משתמשים נכנסים לאתר שלך כדי להוריד רכיב ActiveX, הרכיב עצמו מורד מקובץ CAB המאוחסן באתר, אך שאר קבצי DLL שנדרשים לשם שימוש בו, מגיעים ישירות מ-Microsoft.

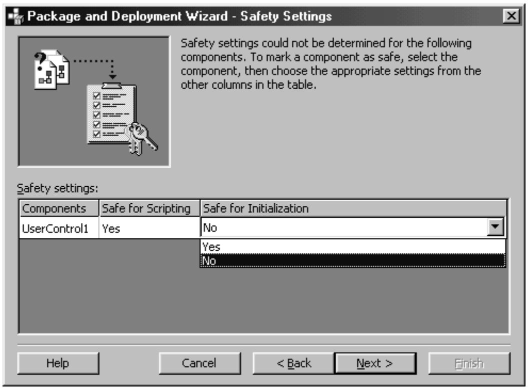


תרשים נספח 2.13: אם יצרת פקד ActiveX המיועד לשימוש ב- Internet Explorer, יש ליצור עבורו Internet Installation Package (חבילה להתקנה מה-Web)

אם החיבור שלך לאינטרנט אינו אמין, איטי או אינו רציף, ייתכן שתרצה לשלב את קבצי Run-time של Visual Basic בקובץ CAB. ייתכן גם שמשיקולי מהירות, תרצה לאחסן את קובץ CAB שבאתר שלך במספר קבצים קטנים. אפשרויות אלו תוכל להגדיר במסך File Source (מקור הקבצים) של האשף אשר מוצג בתרשים נספח 2.14.



תרשים נספח 2.14: תוכל להגדיר מיקום ההורדה של קובץ בלחיצה עליו והזנת כתובת URL או הנתיב



תרשים נספח 2.15: תיבת הדו-שיח Safety Settings (הגדרות אבטחה) משמשת להגדרת רמת האבטחה שאותה יש להגדיר עבור הרכיב במחשב המשתמש

אפשרויות Script

ביצירת חבילת התקנה לרכיב ActiveX המיועד לשימוש בדפי Web חשוב שתוודא שהרכיב יוגדר Safe For Scripting ו-Safe For Initialization, כבתרשים נספח 2.15.

כשרכיב הוא Safe for Scripting ניתן לבצע אליו גישה מקוד שפת Script (כגון VBScript) מבלי להסתכן בגרימת נזק למחשב המשתמש. כשרכיב הוא Safe for Initialization ניתן להגדיר מופע של האובייקט מבלי להזיק למחשב המשתמש. לדוגמה, נניח שיצרת פקד ActiveX שתומך בשיטה למחיקת קבצים. אתרי Web שלך אומנם ינצלו שיטה זו באופן אחראי, אך יש משתמשים שיכתבו קוד משלהם וינצלו שיטה זו באופן מזיק. מובן שכדי שתוכל להשתמש בפקד שיצרת על אפשרויות אלו להיות פעילות, לכן כדאי שתתקיד מחשבה רבה לשיטות ולמאפיינים של הפקדים שאליהם תאפשר למשתמשים גישה.

ראה, "התקנת פקדים באמצעות ה-Web", פרק 14.

ראה, "שימוש ב-VBScript", פרק 30.

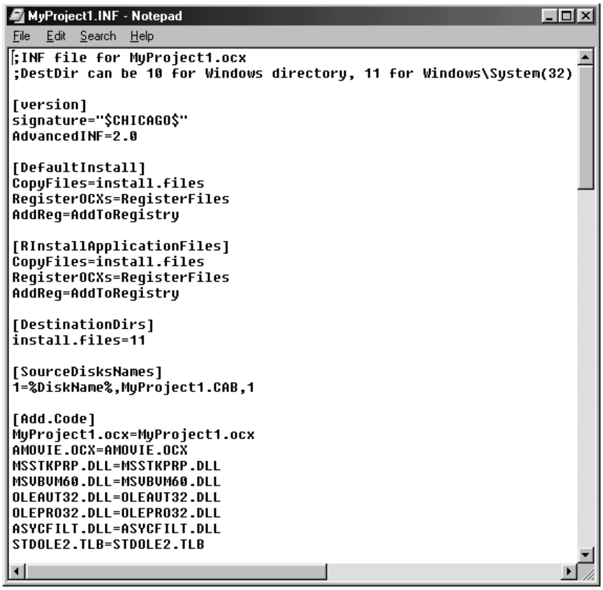
קבצים המיועדים לשימוש באינטרנט

Package and Deployment Wizard יוצר סוג קובץ CAB שונה עבור חבילות להתקנה מהאינטרנט. בוודאי זכור לך שבחבילת התקנה רגילה יש קובץ SETUP.EXE שהמשתמשים נדרשים להפעילו, ואילו חבילה להתקנה מהאינטרנט אינה נזקקת לתוכנית התקנה, מפני שההתקנה מטופלת על ידי Internet Explorer. קובץ CAB מכיל קובץ INF (קיצור של INFormation), המורה ליישום Internet Explorer אלו קבצים להתקין. בתרשים נספח 2.16 תוכל לראות דוגמה לקובץ INF.

בעת יצירת פקד ActiveX, Internet Explorer ייצור גם דוגמת קובץ HTML. קובץ זה יכיל את קוד HTML הנדרש לשם שימוש בפקד בדפי Web :

```
<OBJECT ID="UserControl1"
CLASSID="CLSID:9A15C6AE-CB22-11D1-BE62-10005A75B6DB"
CODEBASE="Project1.CAB#version=1,0,0,0">
</OBJECT>
```

שורות HTML אלו הן כל שנידרש להטבעת אובייקט ActiveX בדף Web. שים לב שיש בקוד פירוט של Class ID הייחודי של האובייקט ושל מספר גרסת האובייקט. Internet Explorer משתמש בנתונים אלה ובנתונים שבקובץ INF, כדי לדעת האם הוא נידרש להוריד את האובייקט ולהטביע אותו. יכולת זו היא אחד היתרונות של השילוב שבין טכנולוגיית ActiveX ו- Internet Explorer. היא מאפשרת לאחסן גרסאות מעודכנות של פקדים באתר Web, תוך ידיעה שגרסאות אלו תותקנה אוטומטית במחשבי המשתמשים.



תרשים נספח 2.16: קובץ INF מורה ל- Internet Explorer אילו קבצים עליו להתקין והיכן יוכל למצוא אותם

מכאן...

בנספח זה חקרת את השימוש באשף Package and Deployment ליצירת תוכנית התקנה עבור משתמשי היישום. למדת מעט גם על אופן הפעולה של תוכנית היישום, ועל פעולות שיש לבצע לצורך התאמה אישית של תוכנית זו. מידע נוסף על נושאים הקשורים להתקנת תוכנות תוכל למצוא בפרקים הבאים:

- מידע נוסף אודות פקדי ActiveX תוכל למצוא בפרק 14 **יצירת פקדי ActiveX**.
- בפרק 31 **מסמכי ActiveX**, תוכל על יצירת מסמכי ActiveX.

נספח 3

תקציר פקודות SQL

מה בפרק?

- הגדרת המושג SQL
- משפטי SELECT
- משפטי פעולה ב-SQL
- משפטי Data-Definition-Language
- שימוש ב-SQL
- בניית משפטי SQL
- אופטימיזציית ביצועים ב-SQL
- העברת משפטי SQL למנגנוני מסדי נתונים אחרים

במספר פרקים קודמים, שעסקו בעבודה עם מסדי נתונים ראית דוגמאות לשימוש במשפטי SQL לצורך הגדרת מערכי רשומות. נספח זה יסביר כיצד ליצור משפטי SQL כאלה, וכיצד לבצע פעולות מתקדמות באמצעות SQL. כל הדוגמאות שתובאנה בפרק תתייחסנה למסדי נתונים של Access, אך הטכניקות שתפורטנה תהיינה ישימות גם לסוגי מסדי נתונים אחרים. ניתן לומר ש-SQL היא אבן היסוד בעבודה עם סוגים רבים של שרתי מסדי נתונים כגון Oracle או SQL Server.

בפרק זה יתוארו שני סוגים עיקריים של משפטי SQL: משפטי Data Manipulation Language (**DML** - משפטים לביצוע פעולות על נתונים) ומשפטי Data Definition Language (**DDL** - משפטי הגדרת נתונים). רוב הפרק יעסוק במשפטי DML ואם לא יצוין אחרת, תוכל להניח שמשפט נתון הוא משפט DML.

הגדרת המושג SQL

Structured Query Language (SQL - שפת שאילתות מובנית) היא מערכת פקודות תכנות אשר מאפשרת למפתח (ולמשתמש הקצה) לבצע פעולות מהסוגים הבאים:

- לאחזר נתונים מטבלה או מטבלאות במסד נתונים או בכמה מסדי נתונים.
- לבצע פעולות על נתונים בטבלאות על ידי הוספה, גריעה או שינוי רשומות.
- להפיק נתוני סיכום כגון: מספרי רשומות, ערכי מינימום, מקסימום וממוצעים על בסיס הנתונים שבטבלאות.
- ליצור, לשנות ולגרוע טבלאות ממסדי נתונים.
- ליצור ולגרוע אינדקסים של טבלאות.

משפטי SQL מאפשרים למפתחים לכתוב משפטי קוד מעטים שיבצעו פעולות שלשם מימושן ב- Visual Basic יידרשו כ- 50-100 שורות קוד.

כיצד פועלת SQL

משם השפה משתמע, שהמשפטים שלה משמשים לבניית שאילתות שמעובדות על ידי מנגנון מסד הנתונים. השאילתה מגדירה את השדות שיש לעבד, הטבלאות שמכילות את אותם שדות, טווח רשומות שאליו היא תתייחס, והגדרות הסדר שבו תוצגנה הרשומות המאוחזרות.

משפטי SQL מחזירים לרוב רשומות במבנה **Dynaset** (מערך רשומות דינמי). בוודאי זכור לך כי Dynaset הוא מערך רשומות בר-עדכון שמכיל למעשה מצביעים לנתונים שבמסד הנתונים. Dynasets אינם קיימים באופן קבוע ולאחר סגירתם אין גישה אליהם. ל-SQL אמצעים לטיפול באחסון קבוע של הרשומות המאוחזרות.

הערה:
תחביר Microsoft SQL המוצג בדוגמאות מיועד לשימוש עם מנגנון Jet **ותואם** לתחביר ANSI SQL (אם כי קיימים הבדלים קטנים בין Microsoft SQL ל- ANSI SQL). **אם** תשתמש ב-SQL להפנית שאילתות לשרת מסד נתונים חיצוני כמו SQL Server או Oracle, כדאי שתעיין בחומר התיעוד של השרת, כדי לוודא באילו יכולות שפת SQL שרת זה תומך, ומהו התחביר המשמש להעברת פקודות אליו.

מרכיבי משפט SQL

משפטי SQL מורכבים משלושה חלקים:

- **הצהרה על פרמטרים** (Parameter Declartions) - אלה פרמטרים אופציונליים המועברים למשפט SQL על ידי התוכנית.
- **פקודת הפעולה** (Manipulative Statement) - חלק המשפט המורה למנגנון מסד הנתונים איזו פעולה לנקוט, לדוגמה SELECT או DELETE.
- **הצהרות על אפשרויות אופציונליות** (Options Declarations) - ההצהרות מורות למנגנון מסד הנתונים על **תנאי סינון** (Filter Conditions), **נתוני קיבוץ** (Data Groupings), **והגדרות מיון** (Sorts), שחלים על הנתונים. בחלק זה ניתן למצוא **קטעי משפט** (Clauses) כמו WHERE, GROUP BY ו-ORDER BY.

סדר החלקים במשפט הוא:

```
[Parameter Declarations] Manipulative Statement [options]
```

בקטע Parameter Declarations תגדיר פרמטרים שישמשו במשפט SQL. כל הערכים שיוגדרו בקטע זה יוצבו במשתנים המתאימים לפני ביצוע המשפט.

ברוב הדוגמאות שתובאנה במהלך הנספח תבנה רק פקודות פעולה והצהרות על אפשרויות אופציונליות. על ידי שימוש בשני חלקי משפט אלה תוכל להגדיר שאילתות שתבצענה מיגוון פעולות רחב. בטבלת נספח 3.1 יפורטו ארבעה סוגים של פקודות פעולה תוך תיאור הפעולות שתבוצענה על ידן.

טבלת נספח 3.1: משפטי הפעולה

משפט	הפעולה שתבוצע על ידי המשפט
DELETE FROM	גריעת רשומות מטבלה
INSERT INTO	הוספת קבוצת רשומות לטבלה
SELECT	אחזור קבוצת רשומות מטבלה ואחסון הרשומות שאוחזרו במערך Dynaset או במערך Table
UPDATE	מגדירה ערכי שדות בטבלה

משפטי הפעולה מורים למנגנון מה לעשות, אך Option Declarations מורות למנגנון אילו שדות ורשומות עליו לעבד. הדיון על Option Declarations יהווה את רובו של פרק זה. במהלך הפרק נסקור את הפרמטרים המשמשים במשפטי SELECT, אחר נשתמש בפרמטרים אלה כחלק ממשפטי פעולה אחרים. רוב הדוגמאות בפרק מתייחסות למסד נתונים לדוגמה בית מסחר לאקווריומים.

במהלך הדיון בסוגים השונים של משפטי SQL נציג רק את תחביר המשפט. חשוב שתהיה מודע לכך שלא ניתן להשתמש במשפטי SQL באופן עצמאי ב- Visual Basic. משפטי SQL משמשים ליצירת אובייקטי QueryDef ומערכי רשומות מסוג Dynaset, או Snapshot על ידי שימוש בשיטה Execute, או כערכים המוצבים במאפיין RecordSource של פקד Data. סעיף זה יתייחס לשימוש ב-SQL ללא תלות בגורמים אחרים, בעוד שהסעיף **שימוש ב-SQL** יתייחס לשימוש ב-SQL בקוד תוכנית.

הערה:

אובייקט QueryDef הוא חלק במסד נתונים שבו מאוחסנת הגדרת שאילתה. ההגדרה היא אותו משפט SQL שחיברת.

משפט Select

משפט SELECT מאחזר רשומות (או שדות מסוימים מתוך רשומות) ומאחסן את הנתונים המוחזרים במערך רשומות מסוג Dynaset או מסוג Table לצורך עיבוד נוסף על ידי התוכנית. תחביר משפט SELECT הוא:

SELECT [predicate] *fieldlist* FROM *Tablelist*
[table relations][range options] [sort options] [group options]

הערה:

בדוגמאות תחביר נקטתי בכללי הכתיב הבאים: מילות מפתח SQL תוצגנה באותיות רישיות, קטעים המוצגים בכתב נטוי הם קטעים שהמפתח יחליף בביטויים אחרים. לדוגמה, את הביטוי Fieldlist ניתן להחליף בביטוי LastName, FirstName. מילים וביטויים המוצגים בסוגריים מרובעים הם אופציונליים.

המרכיבים השונים של המשפט שהוצג לעיל יוסברו במהלך הפרק. משפטי SQL עשויים להיות מורכבים מאוד, אך הם גם עשויים להיות פשוטים מאוד. להלן מוצג המבנה הפשוט ביותר האפשרי עבור משפט SELECT:

```
SELECT * FROM Sales
```

הגדרת השדות הרצויים

קטע fieldlist במשפט SQL משמש להגדרת השדות במערך הרשומות שיווצר כתוצאה מפעולת השאילתה. תוכל לשלב במערך הרשומות את כל השדות שבטבלה מסוימת, שדות נבחרים ואפילו שדות המכילים ערכים מחושבים שיתבססו על ערכים משדות אחרים. את השדות תוכל לבחור מטבלה מסוימת או מתוך כמה טבלאות.

מבנה קטע fieldlist שבמשפט SELECT הוא כדלקמן:

```
[tablename].field1 [AS alt1][,[tablename.]field2 [AS alt2]]
```

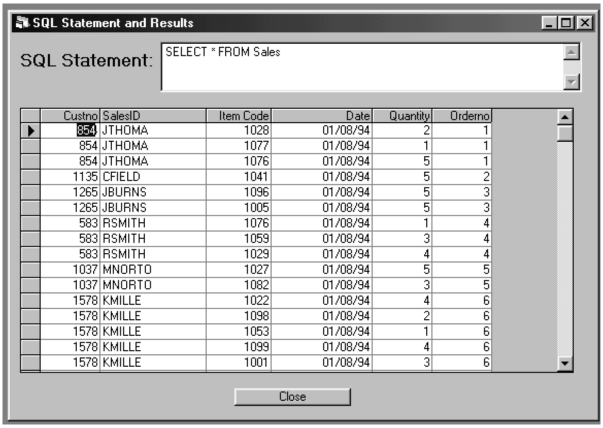
בחירת כל השדות מטבלה

הפרמטר '*' (המכונה גם תו חיפוש כללי Wildcard) משמש כדי להורות למנגנון שאתה מעוניין לבחור את כל השדות שבטבלה מסוימת. תו החיפוש הכללי משמש בקטע fieldlist של המשפט. שימוש במשפט SELECT * FROM Sales בשאילתה שתופעל על מסד הנתונים לדוגמה שתבנה, יצור את מערך הרשומות המוצג בתרשים נספח 3.1.

בחירת שדות מסוימים מטבלה

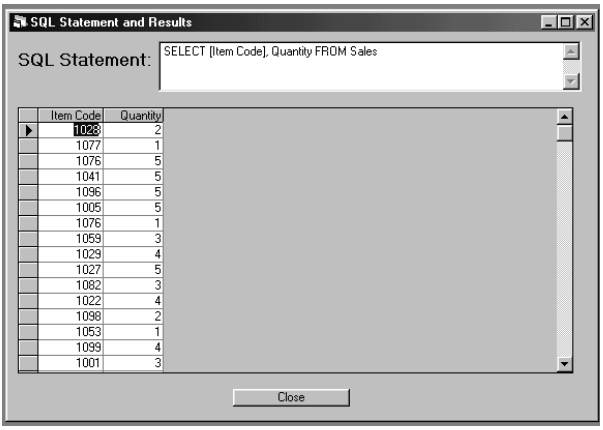
במקרים רבים ידרשו לך רק מספר שדות בטבלה. תוכל להגדיר אילו שדות רצויים על ידי כך שתשלב רשימת שדות במשפט SELECT. הפרד בין השדות באמצעות פסיקים. בנוסף, אם שם שדה שברשימה מכיל רווח, כגון Order Quantity, הקף את שם השדה בסוגריים מרובים. מערך הרשומות שיוחזר כתוצאה משימוש במשפט SQL שיוצג להלן, מוצג בתרשים נספח 3.2. מערך רשומות שיווצר תוך שימוש ברשימת שדות נבחרים יהיה יעיל יותר ממערך רשומות שיווצר על ידי שימוש בתו הבחירה הכללי (*) הן מבחינת גודלו והן מבחינת מהירות יצירתו. כעיקרון כדאי שתתמקד את השאילתות במספר השדות הקטן ביותר שיאפשר להשיג את המטרה שלשמה משמשת התוכנית.

```
SELECT [Item Code], Quantity From Sales
```



Custno	SalesID	Item Code	Date	Quantity	Orderno
854	JTHOMA	1028	01/08/94	2	1
854	JTHOMA	1077	01/08/94	1	1
854	JTHOMA	1076	01/08/94	5	1
1135	CFIELD	1041	01/08/94	5	2
1265	JBURNS	1096	01/08/94	5	3
1265	JBURNS	1005	01/08/94	5	3
583	RSMITH	1076	01/08/94	1	4
583	RSMITH	1059	01/08/94	3	4
583	RSMITH	1029	01/08/94	4	4
1037	MNORTO	1027	01/08/94	5	5
1037	MNORTO	1082	01/08/94	3	5
1578	KMILLE	1022	01/08/94	4	6
1578	KMILLE	1098	01/08/94	2	6
1578	KMILLE	1053	01/08/94	1	6
1578	KMILLE	1099	01/08/94	4	6
1578	KMILLE	1001	01/08/94	3	6

תרשים נספח 3.1: שימוש בתו הבחירה הכללי * בפרמטר fieldlist שבמשפט SELECT מוביל לבחירת כל השדות שבטבלת המקור



תרשים נספח 3.2: מערך זה נוצר על ידי הגדרת שדות ספציפיים במשפט SELECT

בחירת שדות מכמה טבלאות

במהלך הדיון בעיצוב מסדי נתונים, שהובא בפרק 24 ***יסודות מסד הנתונים*** ציינו שנרמול נתונים על ידי אחסונם בטבלאות שונות, מאפשר למנוע מצבי נתונים מיותרים. כשתאחזר נתונים לשם הצגה או שינוי, תרצה להציג את הנתונים הקשורים אליהם והמאוחסנים בטבלאות אחרות. SQL מאפשרת לשלב נתונים מטבלאות שונות במערך רשומות אחד.

כדי לבחור נתונים מכמה טבלאות בבת-אחת יש להגדיר בשאילתה שלושה דברים:

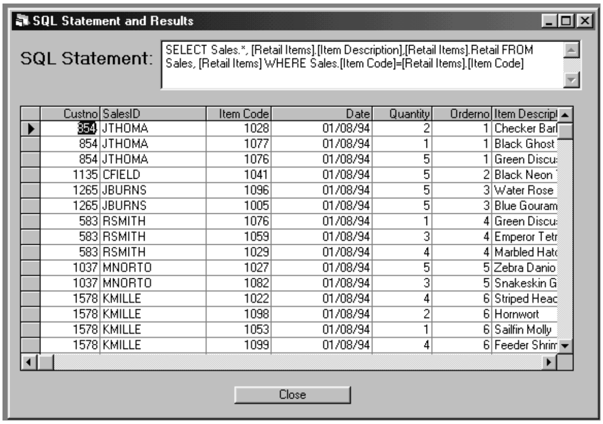
- את הטבלה שממנה ייבחר כל שדה.
- את השדה שמתוכו תבחר את הנתונים.
- קשרים הקיימים בין טבלאות.

הגדר את הטבלה שממנה יילקח כל שדה על ידי כך שתוסיף את שם הטבלה לפני שם השדה כשהוא מופרד ממנו באמצעות נקודה (לדוגמה, Sales.[Item Code] או Sales.Quantity זכור שבעת שימוש בשמות שדות המכילים רווחים, יש להקיף את שמות השדות בסוגריים מרובעים). תוכל גם להשתמש בתו החיפוש הכללי (*) לאחר שם הטבלה כדי לציין שאתה מעוניין לבחור את כל השדות שבטבלה.

הגדר את הטבלאות שאליהן תתייחס השאילתה על ידי כך שתציין את שמותיהן בקטע FROM של המשפט, כשהם מופרדים זה מזה בפסיק.

את הקשרים הקיימים בין הטבלאות תוכל להגדיר בקטע WHERE או בתנאי JOIN של המשפט. קטעי WHERE ו-JOIN יידונו בהמשך הפרק.

משפט SQL המובא בתוכנית נספח 3.1 יאחזר את כל השדות שבטבלה Sales ואת השדות Item Description ו-Retail מטבלת Retail Items. טבלאות אלו מקושרות ביניהן באמצעות שדה Item Code. תוצאות השימוש במשפט זה מוצגות בתרשים נספח 3.3.



תרשים נספח 3.3: בחירה שדות מכמה טבלאות מפיקה מערך רשומות משולב

תוכנית נספח 3.1: SALES.TXT - בחירת שדות מטבלאות באמצעות משפט SQL.

```
SELECT [Retail Items].Retail * Sales.Quantity
FROM Sales, [Retail Items]
WHERE Sales.[Item Code] = [Retail Items].[Item Code]
```

הערה:

ניתן להשמיט שמות טבלאות משאילתות אחזור שדות אם שם השדה בו מדובר מופיע רק בטבלה אחת שברשימת הטבלאות באותה שאילתה. אך הכללת שם הטבלה היא נוהג תכנותי רצוי, שיישומו מקטין את הסיכוי לשגיאות ומשפר את קריאות הקוד.

יצירת שדות מחושבים

הדוגמה המובאת בתוכנית נספח 3.1 מתייחסת לנתוני הזמנות לקוחות ומערך הרשומות המוצג כולל שדות שבהם מוצגים הפריט שהוזמן, הכמות שהוזמנה והמחיר הקמעונאי של הפריט. כדי שתוכל לחשב את העלות הכוללת של הפריטים שהוזמנו השתמש ב**שדה מחושב** (Calculated Field) במשפט SQL. שדה מחושב יכול להכיל תוצאה של פעולה מתמטית שבוצעה על שדות המכילים ערכים מספריים (לדוגמה, Price * Quantity) או פעולת עיבוד מחרוזות שבוצעה על שדות המכילים טקסט (לדוגמה LastName & FirstName). לביצוע פעולות על שדות המכילים ערכים מספריים תוכל להשתמש באופרטורים המתמטיים הרגילים (+, -, *, ., ^), ולביצוע פעולות על שדות המכילים טקסט תוכל להשתמש באופרטור השרשור (&). בנוסף לכך תוכל גם להשתמש בפונקציות Visual Basic לביצוע פעולות על הנתונים המאוחסנים בשדות (תוכל, למשל, להיעזר בפונקציה MID$ כדי לאתר מחרוזת בשדה המכיל טקסט, או בפונקציה UCASE$ כדי להמיר שדה המכיל טקסט לאותיות רישיות, או בפונקציה SQR כדי לחשב את השורש הריבועי של מספר). בתוכנית נספח 3.2 תוכל לראות דוגמאות לשימוש במספר פונקציות בקוד תוכנית.

תוכנית נספח 3.2: TOTPRICE.TXT - **יצירת מיגוון שדות מחושבים בעזרת** SELECT.

If you want to append two string fields together you can use
the ampersand:(&)

SELECT LastName & ', ' & FirstName As FullName FROM Customers

The above statement works in Access. For SQL server, use plus:(+)

SELECT LastName + ', ' + FirstName As FullName FROM Customers

------

Some of the string functions you are used to in Visual Basic are
also available in SQL, as in the following example:

SELECT UCASE$(MID$(LastName,1,3)) & UCASE$(MID$(FirstName,1,3))
AS CustomerID FROM CUSTOMERS

Note again that there are differences between what is acceptable
in MS Access databases and SQL Server. SQL Server does not have
UCASE$ and LCASE$ but instead has UPPER() and LOWER() functions.
For more details see the Transact SQL Help file and search for
'string functions'.

Math functions are also availble:

SELECT Datapoint, SQR(Datapoint) FROM LabData

בקוד שהובא כאן הוגדר שם עבור השדה המחושב. אם לא הוגדר שם, מנגנון הטיפול בשאילתות יקצה לשדה המחושב הראשון שם גנרי כדוגמת Expr1001. בסעיף הבא, **הגדרת שמות חלופיים לשדות** יתואר כיצד תוכל להגדיר שם לשדה.

שדות מחושבים מאוחסנים במערך הרשומות כשדות לקריאה בלבד - כך שלא ניתן לעדכן שדות אלה. פעולות עדכון שתבוצענה על הנתונים שעליהם יתבססו השדות המחושבים לא ישתקפו בערכי השדות.

הערה:

אם תשתמש בשדה מחושב בשילוב עם פקד Data כדאי שתשתמש בפקד Label לשם הצגת תוכן השדה. פעולה כזאת תמנע מהמשתמש לנסות לבצע פעולות לעדכון תוכן השדה, העלולות לגרום לשגיאות. תוכל להשתמש גם בתיבת טקסט שמאפיין Locked (נעילה) שלה **יוגדר** True (מידע נוסף על פקד Data **ועל פקדי איגוד נתונים תוכל למצוא** בפרק 25 **פקד נתונים ופקדי איגוד נתונים**). אם תשתמש בתיבת טקסט, ייתכן שתרצה לשנות את צבע הרקע שלה כאינדיקציה לכך שלא ניתן לשנות את תוכנה.

הגדרת שמות חלופיים לשדות

בתוכנית נספח 3.2 יצרנו שדות מחושבים שייכללו במערך הרשומות. במקרים רבים תרצה להגדיר לשדות כאלה שמות שיהיו שונים משמות ברירת המחדל שמנגנון השאילתות יקצה להם.

ניתן לשנות את התחביר משפט SELECT כדי להגדיר שם עבור שדה מחושב. הקצאת השם נעשית על ידי שימוש ב-AS ובשם הרצוי. אם תרצה תוכל להשתמש בשיטה זו גם לצורך הגדרת שם לשדה רגיל.

תוכנית נספח 3.3: CUSTNAME.TXT - גישה לערך שדה מחושב והגדרת שם השדה.

```
Private Sub Command1_Click()

 'You must have DAO 3.51 referenced to use this code!
 Dim sSQL As String
 Dim db As Database
 Dim NewRS As Recordset
 Dim sPerson As String

 Set db = Workspaces(0).OpenDatabase("D:\BOOK\CODE\C\TEST.MDB")

 'The following code gets a person's name from the database
 'and stores it in the string variable sPerson.
 sSQL = "Select LastName & ', ' & Firstname FROM Customers"
 Set NewRS = db.OpenRecordset(sSQL)
 sPerson = NewRS.Fields(0)

 'The following code does exactly the same thing but references
 'the field in the recordset by its name instead of number.
 sSQL = "Select LastName & ', ' & Firstname As Name FROM Customers"
 Set NewRS = db.OpenRecordset(sSQL)
 sPerson = NewRS.Fields("Name")

 MsgBox sPerson

 'Good programming practice:
 NewRS.Close
 db.Close
 Set NewRS = Nothing
 Set db = Nothing

 'even better programming practice would
 'be to skip this section and use ADO!!:)

End Sub
```

הגדרת מקורות הנתונים

בעת השימוש בשאילתה יש להודיע למנגנון מסד הנתונים מהם הנתונים הדרושים וגם היכן יוכל לאתרם. הודעה זו מועברת בקטע FROM שבמשפט SELECT. המבנה הכללי של קטע FROM הוא:

```
FROM table1 [IN data1] [AS alias1][, table2 [IN data2] [AS alias2]]
```

הסעיפים הבאים ידונו במספר אפשרויות שבהן ניתן להשתמש בקטע FROM.

הגדרת שמות טבלאות

הצורה הפשוטה ביותר של קטע FROM משמשת לפנייה לטבלה יחידה. שורת הקוד הבאה עושה שימוש בצורה זו:

```
SELECT * FROM Sales
```

קטע FROM של המשפט יכול לשמש גם לפנייה למספר טבלאות, בבת-אחת (ראה תוכנית נספח 3.1). בעת פנייה למספר טבלאות בבת-אחת, הפרד בין שמות הטבלאות באמצעות פסיקים. גם כעת יש להקיף שם טבלה המכיל רווח בסוגריים מרובעים (ראה תוכנית נספח 3.1).

שימוש בטבלאות ממסדי נתונים שונים

כשתפתח יישומים רבים יותר, ייתכן שיהיה עליך להתייחס בשאילתה נתונה לטבלאות שבמסדי נתונים שונים. נניח שיש לך מסד נתונים שמכיל נתונים על שם יישוב, שם מדינה ומספר מיקוד. סביר להניח שלא תרצה לשכפל מסד נתונים זה בכל יישום שעשוי להזדקק לו. משפט SELECT מאפשר לאחסן את הנתונים במסד הנתונים המקורי, ולאחזר אותם בכל פעם שיידרשו. לאחזור נתונים ממסד נתונים אחר ממסד הנתונים הפעיל יש לשלב קטע IN בקטע FROM של משפט SELECT. משפט SELECT שישמש לאחזור נתוני המיקוד עם נתוני הלקוחות מוצג בתוכנית נספח 3.4.

תוכנית נספח 3.4: GETCUST.TXT - אחזור נתונים ממספר מסדי נתונים במקביל.

```
SELECT      Customers.LastName,
      Customers.FirstName,
      Zipcode.City,
      Zipcode.State

FROM        Customers,
      Zipcode IN "D:\BOOK\CODE\C\TEST.MDB"

WHERE       Customers.Zip = Zipcode.Zip
```

הגדרת כינוי לשדה בטבלה

שים לב לאופן שבו צוין שם הטבלה בכל שדה שאליו בוצעה פנייה בתוכנית נספח 3.4. משום ששמות שדות עלולים להיות ארוכים, ומשום ששאילתה מסוימת עשויה לפנות לשדות רבים, משפט SELECT עלול להפוך לארוך מאוד. המשפט יהפוך למורכב יותר עם כל שדה וטבלה שתוסיף לו. בנוסף לכך, הצורך להקליד שמות ארוכים מגדיל את הסיכוי לשגיאות הקלדה.

כדי להקל על הבעיה תוכל להגדיר **כינוי** (Alias) לטבלה על ידי שימוש ב-AS בקטע FROM של משפט SELECT. השימוש בקטע AS מאפשר להגדיר שם קצר יותר וייחודי לכל טבלה. תוכל להשתמש בכינוי שתגדיר גם בקטעים אחרים של משפט SELECT, שבהם יש לציין את שם הטבלה שאליה אתה פונה. הקוד המובא בתוכנית נספח 3.5 הוא למעשה שכתוב של הקוד שהובא בתוכנית נספח 3.4, וכולל שימוש בכינויים CS לטבלה Customers ו-ZP לטבלה Zipcode.

תוכנית נספח 3.5: ALIAS.TXT - שימוש בכינוי לטבלה לחיסכון בהקלדה.

```
Using a table alias to cut down on typing:

SELECT      CS.LastName,
        CS.FirstName,
        ZP.City,
        ZP.State

FROMCustomers AS CS,
        Zipcode IN "D:\BOOK\CODE\C\TEST.MDB" AS ZP

WHERE       CS.Zip = ZP.Zip
```

הקטעים DISTINCT ,ALL ו-DISTINCTROW

ברוב היישומים שתבנה, תבחר את כל הרשומות שתענינה על מערכת קריטריונים מסוימת. תוכל לעשות זאת על ידי שילוב קטע ALL לפני רשימת שמות השדות שתיכלל בשאילתה, או על ידי כך שלא תשלב קטעים כלשהם במשפט (ALL מהווה ברירת מחדל). כך ששני משפטי SQL המובאים להלן יבצעו את אותה פעולה:

```
SELECT * FROM Customers
```

```
SELECT ALL FROM Customers
```

אך יהיו גם מקרים שבהם תרצה לזהות את הערכים היחודיים של שדות. במקרים כאלה תוכל להשתמש בקטעים DISCTINCT או DISTINCTROW. קטע DISTINCT מורה למנגנון מסד הנתונים לאחזר רק רשומה אחת המכילה סדרת ערכי שדות מסוימת, ללא תלות במספר הרשומות שבאותה טבלה המכילות את אותה סדרת ערכים רצויה. כדי שרשומה מסוימת תנופה על ידי קטע DISTINCT, כל השדות שלה יידרשו להכיל ערכים זהים לאלה שבשדות מקבילים ברשומה אחרת. לדוגמה, אם תשתמש בקטע

DISTINCT בשאילתה לבחירת שמות משפחה ושמות פרטיים, תוכל לבחור מספר אנשים ששם משפחתם Smith, אך לא תוכל לאחזר יותר מרשומה אחת שבה השם Adam Smith.

כדי לנפות מהטבלה רשומות שהן עותקים מדויקים של רשומות אחרות בטבלה, תוכל להשתמש בקטע DISTINCTROW. DISTINCTROW משווה בין הערכים המאוחסנים בכל השדות ברשומה, אם הם מפורטים בשאילתה ואם לאו. בעת שימוש במסד נתוני הדוגמה שמשמש אותנו במהלך פרק זה, תוכל להיעזר בקטע DISTINCTROW כדי לזהות אילו מוצרים הוזמנו פעם אחת לפחות. לקטע DISTINCTROW לא תהיה השפעה כלשהי בשאילתות שתתייחסנה לטבלה אחת בלבד.

בתוכנית נספח 3.6 מוצגת דוגמה לשימוש בקטעים DISTINCT ו-DISTINCTROW.

תוכנית נספח 3.6: DISTINCT.txt שימוש ב-DISTINCT וב-DISTINCTROW.

```
SELECT DISTINCT [Item Code] FROM Sales

SELECT DISTINCTROW [Retail Items].[Item Code] FROM [Retail Items]
INNER JOIN Sales ON [Retail Items].[Item Code]=Sales.[Item Code]
```

הגדרת קשרים בין טבלאות

כשתגדיר מסד נתונים, תשתמש בשדות מפתח (Key Fields) לקישור בין טבלאות שבמסד הנתונים. לדוגמה, תוכל להשתמש בערך המאוחסן בשדה Salesperson ID שבטבלה Customers לצורך קישור לרשומת איש מכירות שבטבלה Sales Person. פעולות לקישור בין טבלאות מבוצעות כדי לחסוך את הצורך לשלב את נתוני איש המכירות ברשומת כל אחד מהלקוחות. אותם שדות מפתח משמשים גם במשפט SELECT להגדרת קשרים (Relationships) בין טבלאות, שיאפשרו להציג את הנתונים המקשרים בין הטבלאות ולבצע עליהם פעולות. כך תרצה למשל להציג את שם איש המכירות בתצוגת נתוני הלקוח, אך לא את מספר העובד שלו.

קיימים שני סוגים קטעים שמשמשים להגדרת קשרים בין טבלאות:

- **Join** - משלב בין שתי טבלאות בהתאם לתוכן השדות המצוינים בביטוי הצירוף ול**סוג פעולת הצירוף** (Join Type).
- **WHERE** - קטע כזה משמש ברוב המקרים לסינון רשומות שמוחזרות משאילתה, אך קטע WHERE יכול גם לשמש לצורך הדמיית פעולת Inner Join. פעולות Inner Join תתוארנה בסעיף הבא.

הערה:

השימוש בקטע WHERE למיזוג בין טבלאות מוביל ליצירת מערך רשומות לקריאה בלבד. כדי ליצור מערך רשומות ניתן לעדכון, יש להיעזר לשם כך בקטע JOIN.

שימוש בקטע JOIN

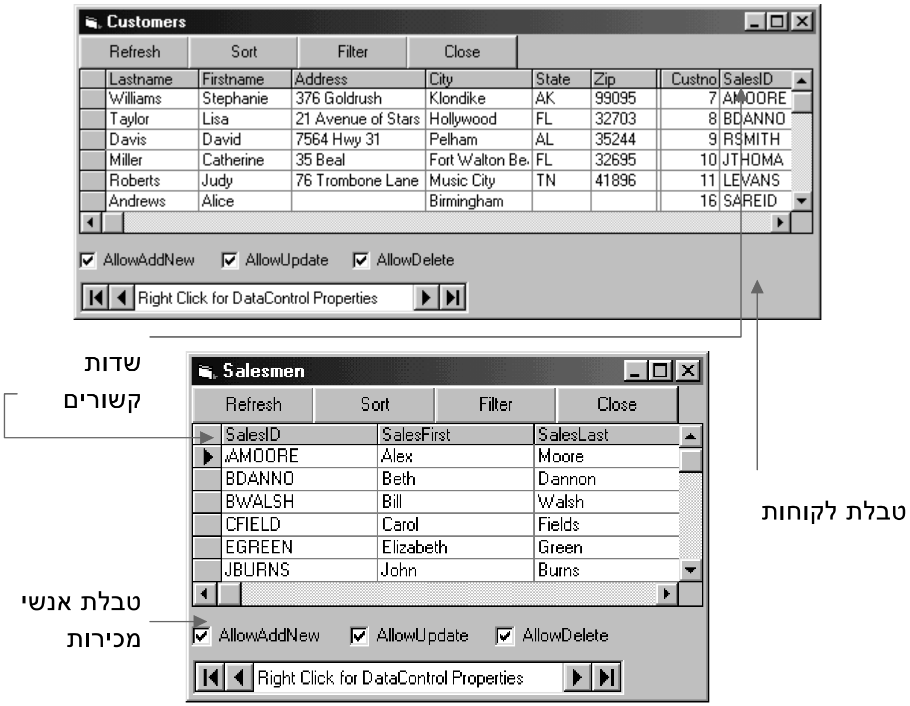
המבנה הבסיסי של קטע JOIN הוא כדלקמן:

```
table1 {INNER|LEFT|RIGHT} JOIN table2 ON table1.key1 = table2.key2
```

מנגנון השאילתות המיושם ב- Visual Basic (וגם ב-Access, Excel ומוצרים נוספים של Microsoft) תומך בשלושה סוגי JOIN: INNER, LEFT ו-RIGHT. כל אחד מהם מחזיר רשומות שעונות על תנאי JOIN, אך כל אחד נוהג באופן שונה במה שקשור להחזרת רשומות שאינן עונות על התנאי. טבלת נספח 3.2 תפרט את הרשומות שתוחזרנה בעת שימוש בכל אחד משלושת סוגי JOIN. בדוגמה הביטוי table1 ״מייצג את הטבלה השמאלית והביטוי table2 ״מייצג את הטבלה הימנית. ככלל, הטבלה השמאלית היא הטבלה הראשונה המאוזכרת בביטוי, והטבלה הימנית היא הטבלה השנייה המוזכרת בביטוי (כלומר שמה מופיע מימין למילת המפתח JOIN).

הערה:

ניתן להיעזר בכל סוגי האופרטורים המשווים (>, >=, =, <=, < ו- <>) בביטוי JOIN.



תרשים נספח 3.4: טבלת Customers וטבלת Salesmen מקושרות בקשר RIGHT JOIN שמאפשר לקשר אנשי מכירות ללקוחות שלהם

טבלת נספח 3.2: הרשומות שתוחזרנה בעת שימוש בסוגי JOIN השונים

סוג JOIN	רשומות מהטבלה השמאלית	רשומות מהטבלה הימנית
INNER	רק רשומות שלהן רשומה תואמת בטבלה הימנית	רק רשומות שלהן רשומה תואמת בטבלה השמאלית
LEFT	כל הרשומות	רק רשומות שלהן רשומה תואמת בטבלה השמאלית
RIGHT	רק רשומות שלהן רשומה תואמת בטבלה הימנית	כל הרשומות

כדי להמחיש את המושגים הרלוונטיים נתייחס למסד הנתונים שבדוגמה שלנו, שמכיל את הטבלאות Customers ו-Salesperson. נניח שהגדרת עשר רשומות של לקוחות וארבע רשומות של אנשי מכירות. שתיים מרשומות הלקוח אינן מקושרות לרשומת איש מכירות ואחת מרשומות אנשי המכירות אינה מקושרת לרשומות לקוחות כלשהם. כל אחת מפעולות JOIN תבחר את אותן שדות, אך יש להגדיר האם אתה מעוניין לבצע פעולת INNER JOIN, LEFT JOIN או RIGHT JOIN (ראה תוכנית נספח 3.7). שתי הטבלאות שאליהן מתייחס קטע הקוד תוצגנה בתרשים נספח 3.4. בתרשים נספח 3.5 יוצגו מערכי הרשומות שיווצרו על ידי כל אחת מפעולות JOIN.

תוכנית נספח 3.7: JOIN.TXT - דוגמאות לשלושת סוגי JOIN.

```
' SELECT USING AN INNER JOIN:
SELECT    CS.LastName,
     CS.FirstName,
     SL.SalesLast,
     SL.SalesFirst

FROMCustomers AS CS
INNER JOIN Salesmen As SL ON CS.SalesID = SL.SalesID

' SELECT USING A LEFT JOIN:
SELECT    CS.LastName,
     CS.FirstName,
     SL.SalesLast,
     SL.SalesFirst

FROMCustomers AS CS
LEFT JOIN Salesmen AS SL ON CS.SalesID = SL.SalesID

' SELECT USING A RIGHT JOIN:
SELECT    CS.LastName,
     CS.FirstName,
     SL.SalesLast,
     SL.SalesFirst

FROMCustomers AS CS
RIGHT JOIN Salesmen AS SL ON CS.SalesID = SL.SalesID
```



תרשים נספח 3.5: סוגי JOIN שונים יחזירו מערכי רשומות שונים

שים לב ש-RIGHT JOIN החזירה את רשומת איש המכירות שאינו מקושר ללקוחות כלשהם, ובנוסף, גם את כל הרשומות שמקושרות לכל אחד מאנשי המכירות האחרים ולא רק רשומה אחת תחת לכל אחד. RIGHT JOIN מיועדת להחזיר את כל הרשומות שבטבלה הימנית, כולל רשומות שאין להן רשומה תואמת בטבלה השמאלית.

שימוש בקטע WHERE

ניתן להשתמש גם בקטע WHERE לקישור בין טבלאות. אופן פעולת קטע WHERE דומה לפעולת INNER JOIN. בתוכנית נספח 3.8 מבוצעת למעשה פעולת INNER JOIN שבוצעה בתוכנית נספח 3.7, אך הפעם הפעולה מבוצעת על ידי קטע WHERE.

תוכנית נספח 3.8: WHERE.TXT - WHERE כפעולה זהה ל- INNER JOIN.

```
' WHERE CLAUSE PERFORMING THE SAME FUNCTION AS AN INNER JOIN
SELECT      CS.LastName,
        CS.FirstName,
        SL.SalesLast,
        SL.SalesFirst
FROMCustomers AS CS,
        Salesmen As SL
WHERE CS.SalesID = SL.SalesID
```

הגדרת קריטריונים לסינון

אחת היכולות החשובות ביותר של משפטי SQL היא אפשרות להגדיר תחום רשומות שעליו תבוצע פעולת עיבוד, על ידי הגדרת **תנאי סינון** (Filter Condition). ניתן להשתמש בסוגי מסנן שונים כגון: Lastname = "Smith",birthday between 5/1/94 and 5/31/94 או Price < 1. הדיון שלנו אומנם יתמקד בשימוש במסננים במשפטי SELECT, אך את העקרונות שיפורטו כאן תוכל ליישם גם בפקודות אחרות של SQL, כגון DELETE ו-UPDATE.

תנאי הסינון במשפטי SQL מצויינים על ידי קטע WHERE. המבנה הכללי הוא:

```
WHERE logical-expression
```

קיימים ארבעה סוגי Predicates (ביטויים לוגיים המשמשים לבדיקת התקיימות תנאי) שבהם ניתן להשתמש בקטע WHER. ביטויים אלה יפורטו בטבלה הבאה:

ביטוי	הפעולה המבוצעת על ידי שימוש בביטוי
Comparison	משווה את הערך המאוחסן בשדה לערך נתון
LIKE	משווה בין הערך המאוחסן בשדה לבין דפוס השוואה כללי (למשל *A)
IN	משווה בין הערך המאוחסן בשדה לבין רשימה ערכים קבילים
BETWEEN	משווה בין הערך המאוחסן בשדה לבין תחום ערכים

שימוש בביטוי השוואה

כמשתמע משמו **ביטוי השוואה** (Comparison Predicate) משמש להשוואה בין ערכי שני ביטויים. SQL תומכת בשישה אופרטורים משווים הניתנים לשימוש בביטויים לבדיקת קיום תנאים. אופרטורים אלה ומשמעויותיהם יפורטו בטבלת נספח 3.3.

המבנה הכללי של משפט בדיקת קיום תנאי עם Comparison הוא:

```
expression1 comparison-operator expression2
```

טבלת נספח 3.3: האופרטורים המשווים המשמשים בקטעי WHERE

אופרטור	משמעות
>	קטן מ-
=>	קטן או שווה ל-
=	שווה ל
>=	גדול או שווה ל
>	גדול מ
<>	שונה מ

כל האופרטורים המשווים דורשים ששני הביטויים שביניהם תתבצע ההשוואה יהיו מאותו הסוג (שני הביטויים יהיו ערכים מספריים, או ששניהם יהיו מחרוזות). בתוכנית נספח 3.9 תובאנה מספר דוגמאות לפעולות השוואה שתתייחסנה לביטויים מסוגים שונים. כדי שיתאפשר לבצע השוואות בין מחרוזות ובין ערכי תאריך יש לבצע פעולות עיצוב מסוימות על הביטויים. מחרוזות שתשמשנה כביטויים משווים יש לתחום בגרשיים (לדוגמה 'Smith' או 'AL') ואילו תאריכים יש לתחום בתווי #, לדוגמה (#15/5/94#). הגרשיים ותווי # משמשים כדי להודיע למנגנון השאילתות מהו סוג הנתון שמועבר אליו. זכור שאין צורך לתחום מספרים בתווים מיוחדים.

תוכנית נספח 3.9: COMPARE.TXT - **אופרטורים משווים ונתונים מסוגים שונים.**

```
Select * From Customers where LastName = 'SMITH'

Select * From [Retail Items] where Retail < 2

Select * From Sales Where Date > #8/15/94#

Note: If you were using SQL server (instead of Jet/Access)
you would place single quotes around the date.
```

שימוש בביטוי הבדיקה LIKE

ביטוי הבדיקה LIKE משמש להשוואה בין ביטוי (כלומר ערך המאוחסן בשדה) לבין דפוס השוואה. הביטוי LIKE מאפשר לאתר שמות משפחה המתחילים באות S, כותרות ספרים המכילות את הביטוי SQL, מילים בנות חמש אותיות שהאות הראשונה היא M והאות האחרונה היא H ועוד. ניתן להשתמש בתווי החיפוש הכלליים '*' וגם '?' בדפוסי ההשוואה. הביטויים שישמשו לאיתור פריטי המידע שתוארו לעיל הם: LastName LIKE 'S*', Title LIKE '*SQL*' ו- Word Like 'M??H', בהתאמה.

מילת ההשוואה LIKE משמשת להשוואה בין מחרוזות בלבד. מבנה ביטוי LIKE הוא:

```
expression LIKE pattern
```

ביטויי LIKE נעזרים בבדיקות התאמה לתווי חיפוש כללים, ובאיתור התאמה לרשימות המכילות תחומי תווים. כשתגדיר דפוס השוואה תוכל לשלב בין תווי חיפוש כלליים ורשימות תווים, כדי לאפשר גמישות בהגדרת הדפוסים. רשימות התווים חייבות לקיים את שלושת התנאים הבאים:

- הרשימה חייבת להיות תחומה בסוגריים מרובעים.
- בין התו הראשון לתו האחרון חייב להפריד מקף.
- תחום התווים חייב להיות מוצג בסדר עולה (לדוגמה A-Z ולא Z-A).

בנוסף, תוכל להציב סימן קריאה (!) לפני הרשימה כדי לאתר תווים שאינם ברשימה. טבלת נספח 3.4 תפרט את ביטויי ההשוואה בהם ניתן להשתמש במשפטי LIKE. תוכנית נספח 3.10 תביא מספר דוגמאות לשימוש בביטויי LIKE במשפט SELECT.

טבלת נספח 3.4: ביטוי LIKE בשילוב עם מבחר דפוסי השוואה

תו חיפוש כללי	משמש לאיתור התאמה ל-	דפוס	תוצאות
*	תווים רבים בבת-אחת	S*	Smith, Sims, sheep
?	תו יחיד	an?	And, ant, any
#	ספרה יחידה	3524#	35242, 35243
[list]	תו יחיד מבין תווים ברשימה	[c-f]	d, e, f
[!list]	תו יחיד שאינו מופיע ברשימה	[!c-f]	a, b, g, h
שילוב תווים	שילוב תווים התואם לדפוס נתון	a?t*	art, antique, artist

תוכנית נספח 3.10: LIKE.TXT - ביטוי LIKE לאיתור מקרי התאמה לדפוס תווים.

```
 ****Multiple character wild card:

SELECT * FROM Customers WHERE Lastname LIKE 'S'*

Note: The above WHERE clause could be read as "Begins with S"
        Changing it to LIKE '*S*' reads as "Contains an S"
Note: On SQL Server, the "*" would be a "%" instead

 ****Single character wild card:

SELECT * FROM Customers WHERE State LIKE '?L'

 ****Character List Matching

SELECT * FROM Customers WHERE MID$(Lastname,1,1) LIKE '[a-f]'
```

ביטוי IN

ביטוי IN מאפשר לבדוק האם ערך ביטוי נתון ברשימת ערכים מסוימת. על ידי שימוש בביטוי IN תוכל למשל לבדוק את קוד המדינה של צרכן מסוים, כדי לוודא האם הוא גר באזור מכירות מסוים. זה המקרה המוצג בדוגמה:

```
SELECT * FROM Customers WHERE State IN ('AL', 'FL', 'GA')
```

ביטוי BETWEEN

ביטוי BETWEEN מאפשר לבצע חיפוש אחר ביטוי שערכו נמצא בתחום ערכים מסוים. הביטוי יכול לשמש לבדיקת מחרוזות, ערכים מספריים וערכי תאריך. החיפוש הוא **חיפוש כולל** (Inclusive Search), כלומר אם הערך שווה לאחת מנקודות קצה התחום,

הרשומה נכללת בחיפוש. ניתן גם להשתמש באופרטור NOT כדי להחזיר ערכים מחוץ לתחום נתון. המבנה הכללי של ביטוי BETWEEN הוא:

```
expression [NOT] BETWEEN value1 AND value2
```

בתוכנית נספח 3.11 תוכל למצוא מספר דוגמאות לשימוש בביטוי BETWEEN.

תוכנית נספח 3.11: BETWEEN.TXT - **שימוש בביטוי** BETWEEN.

```
****String Comparison

SELECT * FROM Customers WHERE Lastname BETWEEN 'M' AND 'W'

****Numeric Comparison

Select * From [Retail Items] where Retail BETWEEN 1 AND 2.5

****Date Comparison

Select * From Sales Where Date BETWEEN #8/1/94# AND #8/10/94#

****Use of the NOT Operator

SELECT * FROM Customers WHERE Lastname NOT BETWEEN 'M' AND 'W'
```

שילוב בין מספר תנאים

WHERE יכול גם להכיל מספר תנאים, כך שתוכל להגדיר תנאי חיפוש שיתבססו על מספר שדות. התנאים שבביטוי המורכב יתבססו על ביטויי הבדיקה שפורטו. קישור בין התנאים יתבצע באמצעות אופרטורים לוגיים מסוג AND או OR. ביטויים מורכבים כאלה יאפשרו למשל לאתר את כל האנשים ששמם Smith וחיים בדרום-מזרח ארצות הברית, או אנשים ששמם הפרטי או שם משפחתם הוא Scott. בתוכנית נספח 3.12 תמצא דוגמאות לתנאי חיפוש אלה. בתרשים נספח 3.6 יוצג מערך רשומות שיוחזר כתוצאה מחיפוש אנשים ששמם הפרטי או שם משפחתם Scott.

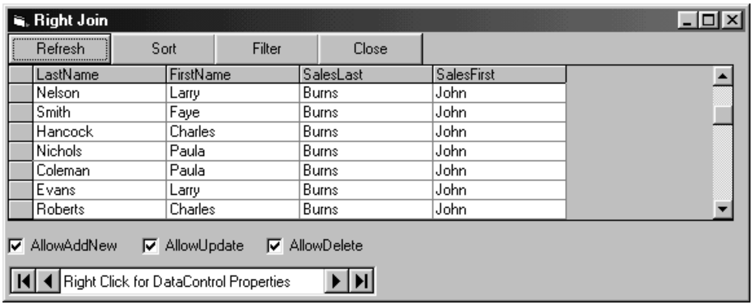
תוכנית נספח 3.12: ANDOR.TXT - **שימוש באופרטורים AND או OR**.

```
Find all the Smiths in the Southeast

SELECT      *
FROM        Customers
WHERE       Lastname = 'SMITH' AND
       State IN ('AL','FL','GA')

Find all Occurences of Scott in first or last name

SELECT
FROM        Customers
WHERE       Lastname = 'SCOTT' OR
       FirstName = 'SCOTT′
```



תרשים נספח 3.6: ניתן לבנות קטעי WHERE משופרים המורכבים ממספר תנאים

הגדרת תנאי המיון

תוכל גם להשתמש במשפט SELECT להגדרת הסדר שבו תוצגנה הרשומות בתוצאת השאילתה. משפט SELECT שולט על הסדר שבו הרשומות תעובדנה ותוצגנה. מיון הרשומות מתבצע על ידי שימוש בקטע ORDER BY.

ניתן להגדיר את סדר המיון על בסיס שדה יחיד או שדות מרובים.

ברירת מחדל של סדר המיון הוא סדר עולה (כלומר A-Z, 0-9), אך ניתן לשנותה על ידי הצבת מילת המפתח DESC (קיצור של DESCENDING - יורד) לאחר שם השדה, בכל שדה שרצוי בו סדר מיון שונה. DESC תשפיע רק על השדה שאליו היא צמודה ולא על שאר השדות שבקטע ORDER BY. בתרשים נספח 3.7 מוצגות תוצאות לשימוש במשפטי SELECT שבתוכנית נספח 3.13.

הערה:

הגדרת אינדקס לשדה המשמש כמפתח מיון עשויה להגביר באופן ניכר את מהירות המיון.

תוכנית נספח 3.13: SORT.TXT - הגדרת סדר המיון של מערך הרשומות שיוחזר

```
' Single Field Sort:

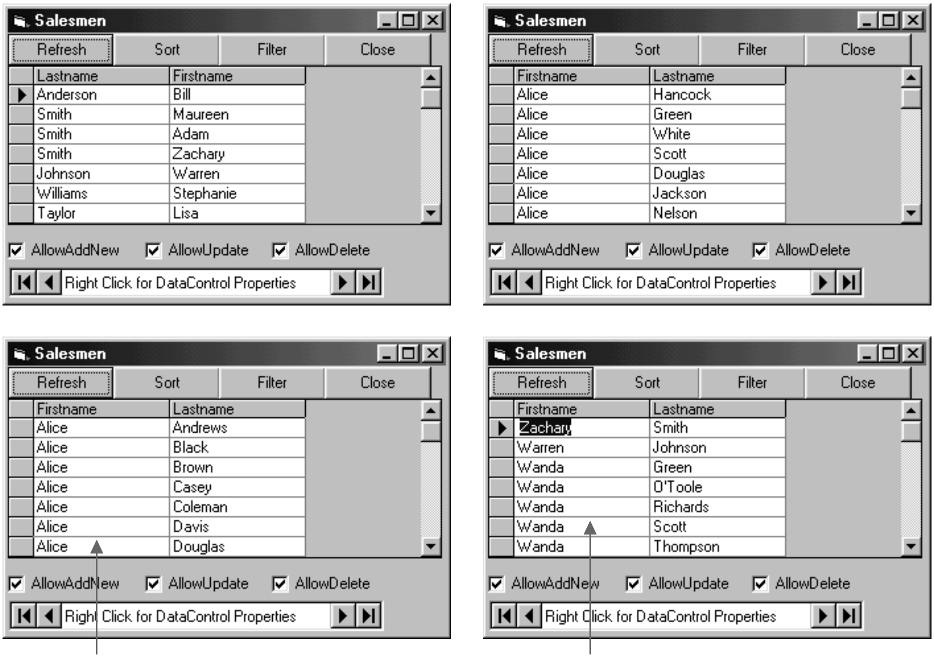
SELECT * FROM Customers ORDER BY LastName

' Multiple Field Sort:

SELECT * FROM Customers ORDER BY LastName, FirstName

' Descending Order Sort:

SELECT * FROM Customers ORDER BY LastName DESC, FirstName
```



מיון בסדר יורד על פי השדה LastName

מיון על פי השדות LastName ו-FirstName

תרשים נספח 3.7: קטע ORDER BY מגדיר את סדר המיון של מערך הרשומות

פונקציות סיכום

ניתן להיעזר במשפטי SELECT לביצוע חישובים על הנתונים שבטבלאות, על ידי שימוש ב**פונקציות צבירה** (Aggregate Functions). את החישובים יש לבצע על ידי הגדרת כשדה במשפט SELECT, תוך שימוש במבנה התחבירי הבא:

function (expression)

הביטוי יכול להתבסס על שדה אחד (SQR (DataPoint)) או על מספר שדות (Quantity * Price). הפונקציה Count יכולה גם להיעזר בתו החיפוש הכללי '*', מפני שהיא מחזירה רק את מספר הרשומות שבשדה הרלוונטי מאוחסן ערך כלשהו. פונקציות הצבירה של SQL מפורטות בטבלת נספח 3.5.

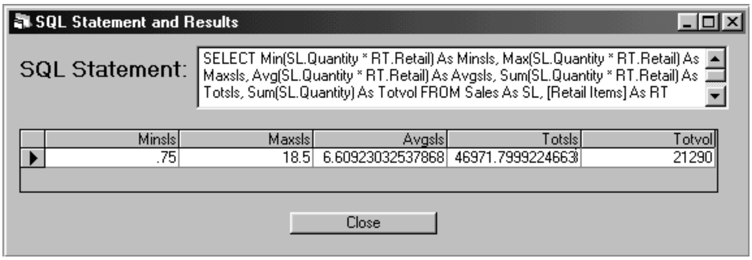
הערה:

מעיין בטבלה שבהמשך נוצר הרושם שהפונקציות StDev ו-StDevP מבצעות פעולה זהה וכך גם הפונקציות Var ו-VarP. ההבדל בין הפונקציות הוא ש-StDev ו-Var מתייחסות לערכי מדגם בעוד StDevP ו-VarP מתייחסות לערכי אוכלוסייה.

טבלת נספח 3.5: פונקציות צבירה מספקות תקציר מידע על הנתונים שבמסד הנתונים

פונקציה	ערך מוחזר
Avg	ממוצע של הערכים שבשדה בכל הרשומות שמקיימות את WHERE
Count	מספר הרשומות שמקיימות את קטע WHERE
Min	הערך הקטן ביותר בשדה מקרב הרשומות שמקיימות את קטע WHERE
Max	הערך הגדול ביותר בשדה מקרב הרשומות המקיימות את קטע WHERE
Sum	סכום הערכים בשדה ברשומות המקיימות את קטע WHERE
First	הערך המאוחסן בשדה ברשומה הראשונה במערך הרשומות
Last	הערך המאוחסן בשדה ברשומה האחרונה במערך הרשומות
StDev	סטיית התקן של הערכים בשדה שברשומות המקיימות את WHERE
StDevP	סטיית התקן של הערכים בשדה שברשומות המקיימות את WHERE
Var	השונות בערכים בשדה המתאים שברשומות המקיימות את קטע WHERE
VarP	השונות בערכים בשדה המתאים שברשומות המקיימות את קטע WHERE

בדומה לפונקציות אחרות של שפת SQL גם פונקציות הצבירה פועלות רק על רשומות שעומדות בתנאי הסינון שמצויינים בקטע WHERE. פונקציות הצבירה אינן מושפעות מסדר מיון הרשומות. הפונקציות מחזירות ערך יחיד המייצג את מערך הרשומות כולו, למעט מקרים בהם נעשה שימוש בקטע GROUP BY, שבהם מוחזר ערך אחד עבור כל קבוצה שמוגדרת על בסיס הביטוי שבקטע. תוכנית נספח 3.14 תציג משפט SELECT המשמש לחישוב ערך מינימלי, ערך מקסימלי, ערך ממוצע, ערך כולל של מכירות מוצרים והכנסות ממכירות כל אחד מסוגי המוצרים. תרשים נספח 3.8 מציג את תוצאת השאילתה.



תרשים נספח 3.8: תקציר הנתונים הנובע משימוש בפונקציות צבירה

תוכנית נספח 3.14: SUMMARY.TXT - שימוש בפונקציות צבירה להצגת תקציר נתונים.

```
SELECT      MIN(SL.Quantity * RT.Retail) As MinSls,
        MAX(SL.Quantity * RT.Retail) As MaxSls,
        AVG(SL.Quantity * RT.Retail) As AvgSls,
        SUM(SL.Quantity * RT.Retail) As TotSls,
        SUM(SL.Quantity) As TotVol

FROM        Sales AS SL,
        [Retail Items] AS RT

WHERE       SL.[Item Code] = RT.[Item Code]
```

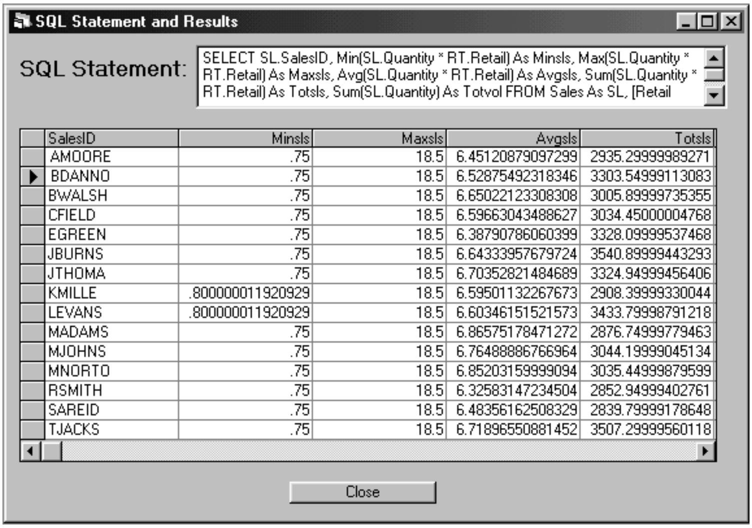
קבוצת רשומות

יצירת **קבוצות רשומות** (Record Groups) מאפשרת ליצור מערכי רשומות שיכילו רשומה אחת בלבד עבור כל מופע של ערך נתון בשדה. לדוגמה, אם תקבץ את הנתונים שבטבלה Customers על פי מדינה, תהיה רשומה אחת שתייצג כל מדינה בטבלה. פעולה זו מועילה במיוחד כשמשלבים אותה עם פונקציות צבירה. שימוש משולב בקבוצות ובפונקציות צבירה יאפשר להציג נתונים כגון סכומי מכירות על פי מדינה, סך המכירות של כל איש מכירות, סך מכירות על פי פריט או סיכומים וחתכים שיתבססו על כל שדה רצוי.

ברוב המקרים תהיה מעוניין להגדיר קבוצות שתתבססנה על שדה יחיד. אך ניתן גם להתייחס לשדות מרובים על ידי קטע GROUP BY. השימוש בקטע זה יוביל ליצירת רשומה נפרדת לכל שילוב ייחודי של ערכי שדות. תוכל להשתמש בשיטה זו לחישוב סכום המכירות של פריטים מסוימים על פי אנשי מכירות. הפרד בין שדות שבקטע GROUP BY באמצעות פסיקים. הקוד בתוכנית נספח 3.15 יציג את תקציר הנתונים שהוצג בתוכנית נספח 3.14 אך הפעם תוך קיבוץ הרשומות על פי מזהי אנשי המכירות. תוצאות השאילתה תוצגנה בתרשים נספח 3.9.

תוכנית נספח 3.15: GROUP.txt חישוב תקציר נתונים של קבוצות רשומות

```
SELECT      MIN(SL.Quantity * RT.Retail) As MinSls,
        MAX(SL.Quantity * RT.Retail) As MaxSls,
        AVG(SL.Quantity * RT.Retail) As AvgSls,
        SUM(SL.Quantity * RT.Retail) As TotSls,
        SUM(SL.Quantity) As TotVol
FROM        Sales AS SL,
        [Retail Items] AS RT
WHERE       SL.[Item Code] = RT.[Item Code]
GROUP BY SL.SalesID
```



SQL Statement and Results

SQL Statement: SELECT SL.SalesID, Min(SL.Quantity * RT.Retail) As Minsls, Max(SL.Quantity * RT.Retail) As Maxsls, Avg(SL.Quantity * RT.Retail) As Avgsls, Sum(SL.Quantity * RT.Retail) As Totsls, Sum(SL.Quantity) As Totvol FROM Sales As SL, [Retail

SalesID	Minsls	Maxsls	Avgsls	Totsls
AMOORE	.75	18.5	6.45120879097299	2935.29999989271
BDANNO	.75	18.5	6.52875492318346	3303.54999113083
BWALSH	.75	18.5	6.65022123308308	3005.89999735355
CFIELD	.75	18.5	6.59663043488627	3034.45000004768
EGREEN	.75	18.5	6.38790786060399	3328.09999537468
JBURNS	.75	18.5	6.64333957679724	3540.89999443293
JTHOMA	.75	18.5	6.70352821484689	3324.94999456406
KMILLE	.800000011920929	18.5	6.59501132267673	2908.39999330044
LEVANS	.800000011920929	18.5	6.60346151521573	3433.79998791218
MADAMS	.75	18.5	6.86575178471272	2876.74999779463
MJOHNS	.75	18.5	6.76488886766964	3044.19999045134
MNORTO	.75	18.5	6.85203159999094	3035.44999879599
RSMITH	.75	18.5	6.32583147234504	2852.94999402761
SAREID	.75	18.5	6.48356162508329	2839.79999178648
TJACKS	.75	18.5	6.71896550881452	3507.29999560118

Close

תרשים נספח 3.9: GROUP BY יאפשר להציג תקציר נתונים עבור כל קבוצה שתגדיר

בקטע GROUP BY ניתן לשלב גם קטע אופציונלי מסוג HAVING. אופן פעולתו דומה לזה של קטע WHERE, אך הוא מתייחס רק לערכי השדות של הרשומות שאוחזרו על ידי השאילתה. קטע HAVING מגדיר אילו מהרשומות הנבחרות תוצגנה, בעוד שקטע WHERE משמש כדי להגדיר אילו רשומות תאוחזרנה מטבלאות הבסיס. בקטע HAVING תוכל להשתמש למשל כדי להציג רק את נתוני אנשי מכירות שסך המכירות החודשי שלהם עולה על $3000. תוכנית נספח 3.16 תציג דוגמה לשימוש בקטע HAVING, ותרשים נספח 3.10 יציג את תוצאת השאילתה שבתוכנית.

תוכנית נספח 3.16: HAVING.TXT - HAVING משמש לסינון הרשומות שתוצגנה

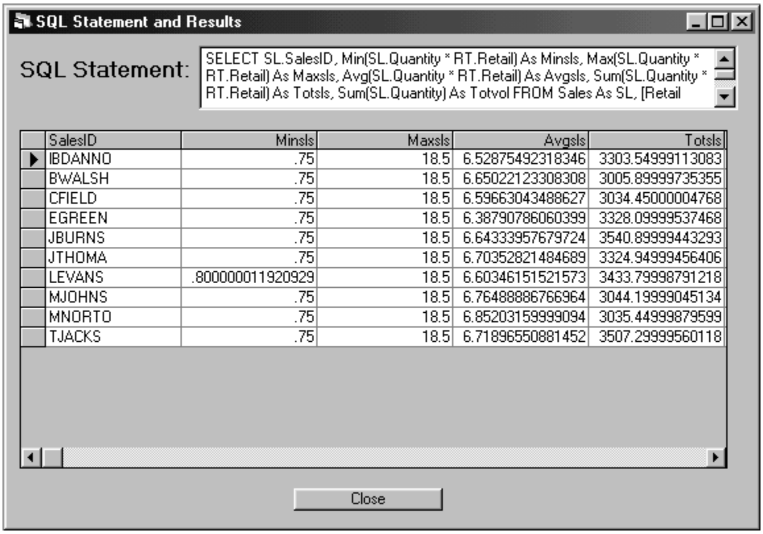
```
SELECT      MIN(SL.Quantity * RT.Retail) As MinSls,
        MAX(SL.Quantity * RT.Retail) As MaxSls,
        AVG(SL.Quantity * RT.Retail) As AvgSls,
        SUM(SL.Quantity * RT.Retail) As TotSls,
        SUM(SL.Quantity) As TotVol

FROM        Sales AS SL,
        [Retail Items] AS RT

WHERE       SL.[Item Code] = RT.[Item Code]

GROUP BY SL.SalesID

HAVING Sum(SL.Quantity * RT.Retail) > 3000
```



תרשים נספח 3.10: קטע HAVING מגביל את תצוגת קבוצת הרשומות

יצירת טבלה

בכל הדוגמאות לשימוש במשפט SELECT שהבאנו עד כה, הרשומות שהוחזרו על ידי השאילתות אוחסנו במערכי רשומות מסוג Dynaset או Snapshot. משום שמערכי רשומות מסוגים אלה הם זמניים, תוכנם קיים רק כל עוד הם פתוחים. לאחר שימוש בשיטה Close או סיום פעולת היישום, מערך הרשומות נעלם (אך השינויים שבוצעו בטבלאות הבסיס נשמרים).

במקרים מסוימים תרצה לשמור את הנתונים באמצעי קבוע לצורך שימוש עתידי. תוכל לעשות זאת על ידי הכללת קטע INTO במשפט SELECT. בקטע INTO תגדיר שם טבלה שבה יישמר פלט השאילתה (ותוכל גם להגדיר שם מסד נתונים שבו תישמר הטבלה). למשל, תוכל ליצור טבלת רשימת דיוור מטבלת לקוחות. תרצה ליצור את רשימת הדיוור באופן שיאפשר להשתמש בה אחר כך באמצעות מעבד תמלילים לביצוע פעולות מיזוג דואר והדפסת תוויות דיוור. הקוד שבתוכנית נספח 3.4 בתחילת הפרק יצר רשימה כזו, שאוחסנה ב-Dynaset. הקוד שבתוכנית נספח 3.17 כולל את אותו משפט SELECT בסיסי ששימש בתוכנית נספח 3.4, אך הפעם נכלל במשפט זה גם קטע INTO שיאחסן בטבלה את הנתונים שיאוחזרו.



אזהרה:

כדאי ששם הטבלה בקטע INTO יהיה שם טבלה חדשה, מפני שאם תשלב בקטע שם טבלה קיימת, היא תידרס על ידי פלט השאילתה.

תוכנית נספח 3.17: INTO.TXT שימוש בקטע INTO לשמירת נתונים בטבלה חדשה

```
' Example of SELECT INTO which creates a new table:

SELECT      CS.FirstName & ', ' & CS.LastName AS FullName,
        CS.Address,
        ZP.City,
        ZP.State,
        CS.ZIP

INTO Mailings

FROMCustomers AS CS,
        Zipcode AS ZP

WHERE      CS.ZIP = ZP.ZIP
```

שימוש בפרמטרים

כל הדוגמאות שהוצגו עד כה כללו התייחסות לערכים ספציפיים. לדוגמה, הגדרנו 'AL' בחיפוש מדינות, או 1.25 בחיפוש מחירים. אך מה אם הערכים אינם ידועים מראש? כאן כדאי להשתמש בפרמטרים. היחס הקיים בין פרמטרים למשפטי SQL הוא כמו זה הקיים בין משתנים לפקודות תוכנית. פרמטר הוא מציין מיקום (Placeholder) שהתוכנית מציבה בו ערך לפני ביצוע השאילתה.

כדי שתוכל להשתמש בפרמטר במשפט SQL, יש להגדיר תחילה את הפרמטר בקטע PARAMETERS במשפט. קטע PARAMETERS יבוא לפני משפט SELECT או כל פקודת פעולה אחרת. בהצהרת הפרמטר יש להצהיר על שמו ועל סוג הנתון שבו. קטע PARAMETERS יופרד מהמשפט בנקודה פסיק (;). לאחר שתצהיר עליו, תוכל להתייחס אליו בקטע הביצועי, ולהשתמש בו כתחליף לערך. הקוד הבא יציג דוגמה:

```
PARAMETERS StateName String SELECT * FROM CUSTOMERS
 WHERE State = StateName
```

כשתבצע משפט זה, התוכנית תתייחס לכל אחד מהפרמטרים כאל מאפיין אובייקט QueryDef. לכן יש להציב ערך בכל משתנה לפני שתקרא לשיטה Execute לצורך ביצוע השאילתה. קטע הקוד הבא ייראה כיצד להגדיר את ערך המאפיין על ידי שימוש במפשט SQL שהוצג לעיל ובמערך רשומות פתוח:

```
Dim OldDb As Database, Qry As QueryDef, Rset As Recordset
Dim OldDb = DBEngine.Workspaces(0).OpenDatabase("C:\Triton.mdb")
Set Qry = OldDb.QueryDefs("StateSelect")
Qry!StateName = "AL"
Set Rset = Qry.OpenRecordset( )
```

בקטע ראית ששימוש בפרמטרים מקל על אחסון שאילתות במסד הנתונים, ומאפשר גמישות משום שהוא מאפשר להגדיר את ערכי ההשוואה בזמן ריצה.

משפטי פעולה

בקטע הקודם ראית כיצד ניתן להשתמש במשפט SELECT לאחזור רשומות ואחסון נתונים ב-Dynaset או בטבלה, כדי לאפשר לבצע עליהם עיבודים נוספים. אך המשפט SELECT היא רק אחד מבין ארבעה משפטים לביצוע פעולות בשפת SQL, שאותם הזכרנו בתחילת הנספח. שלושת המשפטים האחרים הם:

- DELETE FROM - שאילתת פעולה שגורעת רשומות מטבלה.
- INSERT INTO - שאילתת פעולה שמוסיפה קבוצת רשומות לטבלה.
- UPDATE - שאילתת פעולה שמגדירה ערכי שדות בטבלה.

במהלך הסעיפים הבאים נציג אופני שימוש במשפטים אלה לצורך ביצוע פעולות עיבוד נוספות על הנתונים.

משפט DELETE

המשפט DELETE משמש ב**שאילתת פעולה** (Action Query). המשפט מיועד לגרוע רשומות מסוימות מטבלה. שאילתת פעולה אינה מחזירה קבוצת רשומות ב-Dynaset כפי שעושה שאילתת SELECT. במקום זאת היא מתפקדת למעשה כשיגרה המבצעת סדרת פעולות מסוימת וחוזרת לשורה הבאה בתהליך שקרא להן.

תחביר משפט DELETE הוא:

```
DELETE FROM tablename [WHERE clause]
```

קטע WHERE הוא פרמטר אופציונלי. משפט DELETE שלא יכיל קטע WHERE יגרע את כל הרשומות בטבלה שאליה תתייחס השאילתה. ניתן להשתמש בקטע WHERE כדי להגביל את פעולת הגריעה לרשומות העונות על קריטריונים מסוימים. תוכל לשלב בקטע WHERE את כל סוגי האופרטורים המשווים שפורטו בסעיף הדן בביטויי השוואה. להלן דוגמה למשפט DELETE שיגרע מהטבלה את כל הלקוחות המתגוררים בפלורידה:

```
DELETE FROM Customers WHERE State='FL'
```



אזהרה:

משפט DELETE גורם למחיקת הרשומות באופן קבוע ובלתי ניתן לשחזור. המצב היחיד בו אפשר לבצע שחזור הוא בעת שימוש בעיבוד טרנזקציות (Transaction Processing). אם תבצע עיבוד טרנזקציות, תוכל להשתמש בפקודת ROLLBACK כדי לשחזר את כל הרשומות שנגרעו מאז פקודת BEGINTRANS האחרונה.

משפט INSERT

בדומה ל-DELETE גם משפט INSERT הוא סוג של שאילתת פעולה. הואד משמש בשילוב עם משפט SELECT להוספת קבוצת רשומות לטבלה. תחביר המשפט הוא:

```
INSERT INTO tablename SELECT rest-of-select-statement
```

קטע SELECT שבמשפט, יהיה בנוי לפי הכללים שפורטו בסעיף הדן במשפטי SELECT שבתחילת הפרק. קטע SELECT משמש להגדרת הרשומות שתוספנה לטבלה. משפט INSERT יגדיר את פעולת ההוספה, תוך ציון הטבלאות שאליהן תוספנה הרשומות.

אחד השימושים העיקריים של משפטי SELECT הוא לעדכון טבלאות שנוצרו על ידי משפטי SELECT INTO. נניח שמונית לנהל את רשימת החברים במועדון הברידג' שלך. בתחילה יצרת רשימה ראשונית, אך מדי חודש מתוספים חברים חדשים. תוכל לעדכן את הטבלה באחד משני אופנים: על ידי ביצוע חוזר של שאילתת SELECT INTO ששימשה ליצירת הטבלה, או על ידי שימוש בשאילתת INSERT INTO להוספת חברים חדשים לרשימה הקיימת. תוכנית נספח 3.18 תציג דוגמה ליצירת רשימה התחלתית, ושימוש בשאילתה INSERT INTO לעדכון הרשימה.

תוכנית נספח 3.18: INSERT.TXT - INSERT INTO להוספת קבוצה רשומות לטבלה

```
' Example of SELECT INTO which creates a new Mailing list table:
SELECT      CS.FirstName & ', ' & CS.LastName AS FullName,
        CS.Address,
        ZP.City,
        ZP.State,
        CS.ZIP

INTO Mailings

FROMCustomers AS CS,
        Zipcode AS ZP

WHERE       CS.ZIP = ZP.ZIP

' Update the mailing list each month

INSERT INTO Mailings

SELECT      CS.FirstName & ', ' & CS.LastName AS FullName,
        CS.Address,
        ZP.City,
        ZP.State,
        CS.ZIP
FROMCustomers AS CS,
        Zipcode AS ZP
WHERE       CS.ZIP = ZP.ZIP
AND   CS.MemDate > LastMonth
```

משפט UPDATE

המשפט UPDATE היא סוג נוסף של שאילתת פעולה. המשפט משנה את הערכים המאוחסנים בשדות מסוימים בטבלה. תחביר המשפט UPDATE הוא כדלקמן:

```
UPDATE tablename SET field = newvalue [WHERE clause]
```

ניתן לעדכן מספר שדות בבת-אחת על-ידי שילוב מספר קטעי *field=newvalue* במשפט, כשהם מופרדים זה מזה בפסיק. קטע WHERE הוא אופציונלי. משפט UPDATE שלא יכלול קטע WHERE יעדכן את כל הרשומות בטבלה.

הקוד שבתוכנית נספח 3.19 כולל שתי דוגמאות לשימוש במשפט UPDATE. הפקודה הראשונה תשנה את מזהה (ID) איש מכירות ברשומות לקוחותיו, מצב שעשוי לקרות כשאיש מכירות מסוים עוזב את החברה ולקוחותיו מועברים לטיפול איש מכירות אחר. הדוגמה השנייה תעדכן מחיר קמעונאי של פריטים, לאחר עליית מחירים.

תוכנית נספח 3.19: UPDATE.TXT - UPDATE לשינוי ערך במספר רשומות

```
' Change the SalesID for a group of customers

UPDATE Customers SET SalesID = 'EGREEN' WHERE SalesID = 'JBURNS'

' Increase the price of all retail items by five percent

UPDATE [Retail Items] Set Retail = Retail * 1.05
```

משפטי שפת הגדרת נתונים

משפטי Data-Definition-Language (DDL - שפת הגדרת נתונים) מאפשרים ליצור, לשנות, ולגרוע טבלאות ואינדקסים ממסד נתונים במשפט אחד. במצבים רבים משפטים כאלה עשויים לשמש במקום שיטות אובייקטי Data Access Objects שתוארו בפרק 26. אך השימוש במשפטי DDL נתון להגבלות מסוימות. ההגבלה העיקרית היא שמשפטים כאלה יכולים לשמש אך ורק לעבודה עם מסדי נתונים מסוג Jet (בעוד Data Access Objects יכולים לשמש לעבודה עם כל סוג מסד נתונים שמנגנון Jet תומך בהתקשרות אליו). הגבלה נוספת היא העובדה שהם תומכים רק בחלק קטן של מאפייני אובייקטים מסוג Table, field, Index. כדי להתייחס למאפיינים שאינם נתמכים, יש ליישם את השיטות שתוארו בפרק 26.

הגדרת טבלאות באמצעות משפטי DDL

יש שלושה משפטי DDL המשמשים להגדרת טבלאות במסד נתונים :

- CREATE TABLE - מגדיר טבלה חדשה במסד נתונים.
- ALTER TABLE - משנה את הגדרת הטבלה.
- DROP TABLE - גורע טבלה ממסד נתונים.

יצירת טבלה באמצעות משפטי DDL

ליצירת טבלה על ידי משפט DDL, יש ליצור משפט SQL שיכיל את שם הטבלה, שמות השדות שאתה מעוניין לשלב בטבלה, סוגי הנתונים שיאוחסנו בהם וגודלם. קטע הקוד הבא ידגים כיצד ליצור טבלת הזמנות עבור מסד הנתונים לדוגמה שלנו :

```
CREATE TABLE Orders(Orderno LONG, Custno LONG, SalesID TEXT (6),
        OrderDate DATE, Totcost SINGLE)
```

שים לב שאין צורך לתחום את שם הטבלה ושמות השדות במרכאות, אך אם תרצה להגדיר שם טבלה או שם שדה שיכיל רווח, יש לתחום אותו בסוגריים מרובעים (לדוגמה [Last Name]).

במשפט יצירת הטבלה ניתן להגדיר רק שמות שדות, סוגי נתונים שיאוחסנו בשדות וגודלי שדות. לא ניתן לשלב פרמטרים אופציונליים כגון ערכי ברירת מחדל, כללי אימות (Validation Rules) או הודעות שגיאה שתוצגנה במקרי עבירה על חוקי האימות. אך למרות המגבלות, משפט CREATE TABLE היא כלי רב עוצמה שתוכל להשתמש בו ליצירת טבלאות רבות במסדי נתונים.

הכנסת שינויים בטבלה

על ידי שימוש בפקודה ALTER TABLE תוכל להוסיף שדה או לגרוע שדה מטבלה קיימת. בעת הוספת שדה יש להגדיר שם, סוג נתון וגודל (בעת הגדרת שדות הדורשים הגדרה כזו). הוסף את השדה על ידי הכללת קטע ADD COLUMN במשפט ALTER TABLE. גריעת שדות נעשית על ידי ציון שמות השדות בקטע DROP COLUMN. לא תוכל לגרוע שדות שבאינדקס או בקשר בין טבלאות (Relation). הקוד שבתוכנית נספח 3.20 כולל דוגמאות להוספה וגריעת שדה מהטבלה שהגדרת בסעיף הקודם.

תוכנית נספח 3.20: ALTERTABLE - ALTERTAB.TXT להוספה או גריעת שדה

```
'Add a shipping charges field to the "Orders" table

ALTER TABLE Mailings ADD COLUMN Shipping SINGLE

' Delete the shipping charges field

ALTER TABLE Orders DROP COLUMN Shipping
```

גריעת שדה

תוכל לגרוע טבלה ממסד נתונים על ידי שימוש במשפט DROP TABLE. שורת הקוד שלהלן תציג כיצד לגרוע את הטבלה Orders ממסד הנתונים. כדאי לנהוג בזהירות בעת גריעת טבלאות מפני שביצוע המשפט יוביל למחיקה בלתי הפיכה של הטבלה.

```
DROP TABLE Orders
```

הגדרת אינדקסים באמצעות משפטי DDL

יש שני משפטי DDL המיועדים במיוחד לעבודה עם אינדקסים:

- CREATE INDEX - מגדיר אינדקס חדש עבור טבלה.
- DROP INDEX - גורע אינדקס מטבלה.

יצירת אינדקס

המשפט CREATE INDEX מאפשר ליצור אינדקס המבוסס על שדה יחיד או שדות מרובים. הפרמטרים הדרושים ליצירת האינדקס הם: שם האינדקס, שם הטבלה שעבורה יוגדר האינדקס ושם שדה אחד לפחות שעליו יתבסס האינדקס. תוכל גם להגדיר אם האינדקס ימוין בסדר עולה או יורד והאם הוא יהווה אינדקס ראשי. תוכנית נספח 3.21 תציג יצירת אינדקס ראשי שיתבסס על מספר הלקוח, ויצירת אינדקס שיתבסס על שני שדות שבהגדרתו תכלול גם הגדרת סדרי מיון עבור השדות. האינדקסים שיוגדרו בדוגמה יתייחסו לטבלת Customer של מסד הנתונים לדוגמה.

תוכנית נספח 3.21: CREATE INDEX - CREATEIND.TXT ליצירת סוגי אינדקס

```
' Create a primary index on Customer Number

CREATE Index CustNo ON Customers(CustNo) WITH PRIMARY

' Create a two field index with ascending order on LastName and
' descending order on firstname

CREATE Index Name2 ON Customers(LastName ASC, FirstName DESC)
```

מחיקת אינדקס

מחיקת אינדקס פשוטה כמעט כמו יצירתו. לגריעת אינדקס מטבלה תוכל להשתמש במשפט DROP INDEX, כמתואר בדוגמה הבאה. המשפטים שבדוגמה יגרעו מהטבלה את שני האינדקסים שנוצרו על ידי קטע הקוד שבתוכנית נספח 3.21. שים לב שיש לשלב במשפט גם את שם הטבלה שעבורה הוגדר האינדקס הרצוי.

```
DROP INDEX Custno ON Customers
DROP INDEX Name2 ON Customers
```

שימוש ב-SQL

בתחילת הפרק ציינו שלא ניתן לשלב משפט SQL כפקודה עצמאית בתוכנית Visual Basic. משפט SQL חייב להופיע כחלק מפונקציה אחרת. חלק זה של הפרק יוקדש לתיאור השיטות השונות המיושמות לצורך שימוש במשפטי SQL.

ביצוע שאילתת פעולה

מנגנון Jet מציע את השיטה Execute כחלק מאובייקט Database. השיטה Execute מורה למנגנון להפעיל שאילתת SQL על מסד נתונים מסוים. ניתן גם להשתמש בשאילתת פעולה ליצירת אובייקט QueryDef. הגדרת אובייקט QueryDef מאפשרת לבצע את השאילתה מבלי שתהיה תלויה באובייקטים אחרים. קטע הקוד שבתוכנית נספח 3.22 ידגים שימוש בשתי שיטות אלו להפעלת אותו משפט SQL.

תוכנית נספח 3.22: EXECUTE.TXT - ביצוע משפטי SQL על ידי שימוש בשיטות DatabaseExecute ו- QueryExecute.

```
Private Sub Command1_Click()

 'You must have DAO 3.51 referenced to use this code!
 Dim sSQL As String
 Dim db As Database
 Dim NewQry As QueryDef

 Set db = Workspaces(0).OpenDatabase("D:\BOOK\CODE\C\TEST.MDB")

 sSQL = "UPDATE Customers SET SalesID = 'EGREEN'"
 sSQL = sSQL & "WHERE SalesID = 'JBURNS'"

 'The following line of code executes the update statement
 db.Execute sSQL

 'The following code does exactly the same thing using a QueryDef
 Set NewQry = db.CreateQueryDef("Change Sales", sSQL)
 NewQry.Execute

 'Or, you can execute the QueryDef by name if you prefer:
 db.Execute "Change Sales"

  'Good programming practice:
 NewQry.Close
 db.Close
 Set NewQry = Nothing
 Set db = Nothing

 'even better programming practice would
 'be to skip this section and use ADO(!!:)

End Sub
```

יצירת אובייקט QueryDef

על ידי יצירת אובייקט QueryDef תוכל להגדיר שם שאילתה ולאחסן אותה במסד הנתונים עם הטבלאות. תוכל להגדיר שאילתת פעולה או **שאילתת אחזור נתונים** (Retrieval Query) ולאחר שתיצור את השאילתה, תוכל לבצע אותה בקריאה בשמה (פעולה כזו תוארה בקטע הקוד שהוצג בסעיף הדן על ביצוע שאילתת פעולה). בתוכנית נספח 3.22 הוצגה דוגמת ביצוע גישה לאובייקט QueryDef בשם Change Sales לצורך עדכון מזהה איש המכירות.

יצירת Dynasets ו-Snapshots

כדי שתוכל להיעזר במשפט SELECT לאחזור רשומות ואחסונן במערך רשומות מסוג Dynaset או Snapshot, יש להשתמש במשפט SELECT בשילוב עם השיטה OpenRecordset. בעזרת השיטה OpenRecordset תוכל להגדיר את סוג מערך הרשומות שאתה מעונין לפתוח, ואת האפשרויות שאתה מעוניין ליישם לצורך פתיחתו. בעת שימוש בשיטה OpenRecordset תוכל להשתמש במשפט SELECT באופן ישיר, או שלחילופין תוכל להשתמש בשם שאילתת אחזור שהגדרת קודם לכן. תוכנית נספח 3.23 תציג דוגמאות לשימוש בשני אופנים אלה.

תוכנית נספח 3.23: CREATEMETH.TXT - שימוש בשיטות Create לאחזור רשומות שהוגדרו במשפט Select

```
Private Sub Command1_Click()

 'You must have DAO 3.51 referenced to use this code!
 Dim sSQL As String
 Dim db As Database
 Dim NewQry As QueryDef
 Dim NewRS As Recordset

 Set db = Workspaces(0).OpenDatabase("D:\BOOK\CODE\C\TEST.MDB")

 sSQL = "Select RI.[Item Description], SL.Quantity, RI.Retail",
 sSQL = sSQL & "SL.Quantity * RI.Retail As SubTot"
 sSQL = sSQL & "FROM [Retail Items] AS RI, Sales AS SL"
 sSQL = sSQL & "WHERE SL.[Item Code] = RI.[Item Code]"

 'Create the Recordset directly
 Set NewRS = db.OpenRecordset(sSQL, dbOpenDynaset)

 'Or, you can use a QueryDef
 Set NewQry = db.CreateQueryDef("Get Subtotals", sSQL)
 NewQry.Close

 Set NewRS = db.OpenRecordset("Get Subtotals", dbOpenSnapshot)
```

```
'Good programming practice:
NewRS.Close
db.Close
Set rs = Nothing
Set NewQry = Nothing
Set db = Nothing

'even better programming practice would
'be to skip this section and use ADO(!!:)

End Sub
```

ראית כיצד ניתן להשתמש במשפטי SELECT ליצירת מערכי רשומות מסוג Dynaset או Snapshot. אך קטע ההשוואה של WHERE ורשימת הגדרות המיון בקטע ORDER BY יכולים לשמש גם להגדרת המאפיינים של אובייקטי Dynaset. מאפיין Dynaset Filter הוא למעשה משפט WHERE שאינו כולל את מילת המפתח WHERE. בעת הגדרת מאפיין Filter תוכל להשתמש בכל ביטויי הבדיקה שפורטו בקטע הדן על שימוש בקטע WHERE. גם מאפיין Sort של אובייקט Dynaset הוא למעשה פקודת Sort שאינה כוללת שימוש במילות המפתח ORDER BY.

משפטי SQL בשילוב עם פקד Data

פקד Data משתמש במאפיין RecordSource ליצירת מערך רשומות בעת שהוא נטען. הערך המוצב במאפיין RecordSource עשוי להיות שם טבלה, משפט SELECT או שם שאילתה שהוגדרה קודם לכן. על-כן ניתן לומר שכל מה שנאמר על משפט SELECT, רלוונטי גם ליצירת מערכי רשומות שישמשו בשילוב עם פקדי Data.

הערה:

בהצבת שם טבלה במאפיין RecordSource של הפקד, Visual Basic נעזרת בשם זה ליצירת משפט SELECT כמו זה המוצג להלן:

```
SELECT * FROM table
```

יצירת משפטי SQL

כשתרצה ליצור משפטי SQL ולנסותם, תוכל לכתוב אותם ישירות בקוד התוכנית ולהפעילו כדי לוודא שהם פועלים באופן הרצוי. אך זהו תהליך גוזל זמן ועלול להיות מתסכל מאוד, בעיקר כשיש לנסות משפטים מורכבים מאוד. אך מוצעים לך גם שלושה כלים אחרים לפיתוח משפטי SQL שהשימוש בהם עשוי להיות פשוט יותר:

- התוספת Visual Data Manager שב- Visual Basic.
- תוכנת Access (אם מצוי ברשותך עותק התוכנה).
- Microsoft Query.

הערה:

משתמשי Excel או Office יכולים גם להשתמש בכלי Microsoft Query של Access.

Visual Data Manager ו-Access כוללים כלים לבניית שאילתות שמסייעים בבניית שאילתות SQL. הכלים כוללים תיבות דו-שיח המסייעות בבחירת השדות שייכללו בשאילתה, ואמצעים שמסייעים בניסוח הקטעים. לאחר שתסיים לנסות שאילתה באמצעות אחד הכלים, תוכל לאחסן אותה במסד הנתונים כאובייקט QueryDef. תוכל לבצע את השאילתה בתוכנית בקריאת שמה. לחילופין, תוכל להעתיק את הקוד מכלי בניית השאילתה לתוכנית, על ידי פעולות חיתוך והדבקה רגילות.

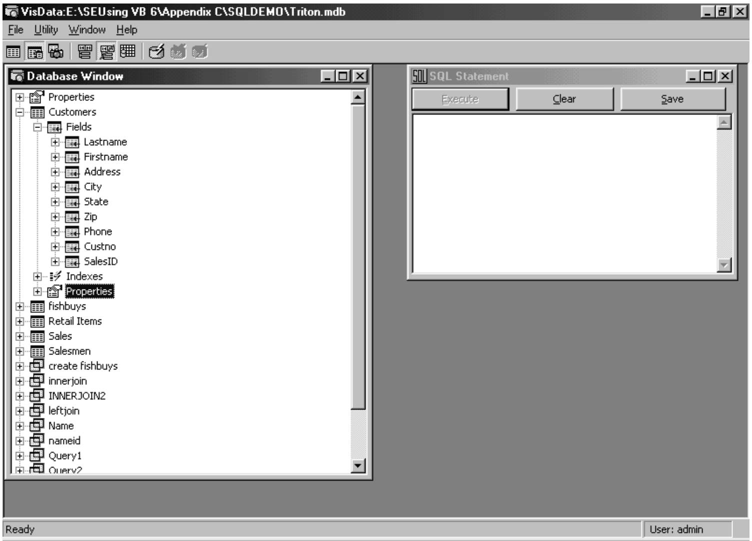
Visual Data Manager

Visual Data Manager הוא תוספת (Add-In) המאפשרת ליצור ולשנות מסדי נתונים שישמשו בתוכניות הכתובות ב- Visual Basic. ל- Visual Data Manager יש גם חלון שמאפשר להזין שאילתות SQL ולנפות מהן שגיאות. אם לא תרצה להשקיע את המאמץ הדרוש ליצירת השאילתה, VDM יציע בונה שאילתות (Query Builder), שיאפשר לבנות שאילתות על ידי בחירת פריטים שהוא מציע.

הערה:

אם תרצה ללמוד אודות אופן הפעולה של Visual Data Manager כדאי שתדע שהוא למעשה אחד הפרויקטים לדוגמה שמסופקים ב- Visual Basic. שם קובץ הפרויקט הוא VISDATA.VBP וניתן למצוא אותו בתיקיה VISDATA שבתיקיה Samples.

הפעל את Visual Data Manager על ידי בחירה בפקודה Visual Data Manager מתפריט Add-Ins של Visual Basic. פתח את תפריט File של התוסף ובחר Open Database. אחר בחר את סוג מסד הנתונים הרצוי מתפריט המשנה. תוצג תיבת דו-שיח שתאפשר לפתוח את מסד נתונים. בשלב זה תוצג בחלונית השמאלית של חלון היישום רשימת הטבלאות והשאילתות שהוגדרו במסד הנתונים. Visual Data Manager מוצג בתרשים נספח 3.11 בעת עבודה עם מסד הנתונים Triton.mdb.

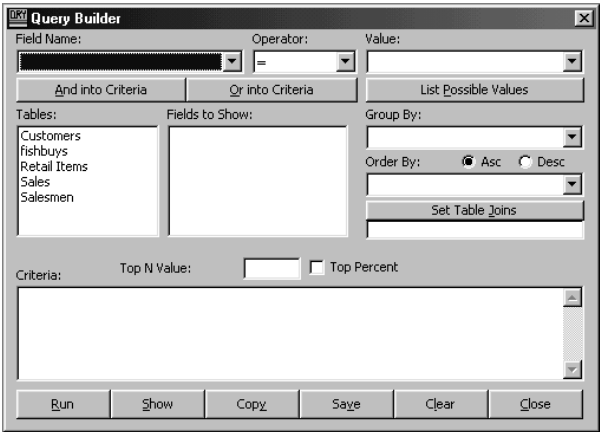


תרשים נספח 3.11: תוכל להיעזר ב- Visual Data Manager לפיתוח שאילתות SQL

כשתרצה לפתח שאילתה ולנסות אותה, יש להקליד תחילה את משפט SQL בתיבת הטקסט שבתיבת הדו-שיח SQL Statement (תיבת הדו-שיח הימנית שבתרשים נספח 3.11) כשתהיה מוכן לבדוק אותו, לחץ על לחצן Execute. אם השאילתה שיצרת היא שאילתת אחזור, ביצוע השאילתה ייצור Dynaset והרשומות שתוחזרנה על ידי השאילתה תוצגנה בטופס הזנת נתונים, (או Grid). אם השאילתה היא שאילתת פעולה, תוצג תיבת הודעה שתודיע על השלמת הפעולה (בהנחה שלא אותרו שגיאות). אם משפט SQL מכיל שגיאה כלשהי, תוצג תיבת הודעה שתדווח עליה.

Visual Data Manager כולל גם בונה שאילתות. תוכל להציג אותו על ידי בחירה בפקודה Query Builder מתפריט Visual Data Manager Utilities. לבניית שאילתה על ידי שימוש בכלי Query Builder, יש לבצע את הצעדים הבאים:

1. בחר את הטבלאות שתרצה לשלב בשאילתה מתיבת הרשימה **Tables**.
2. בחר את השדות שאליהם תתייחס בשאילתה מתיבת הרשימה **Fields to Show**.
3. הגדר את קטע WHERE, אם יש כזה, על ידי שימוש בתיבות הרשימה הנפתחות **Field Name**, **Operator** או **Value** המוצגות בראש תיבת הדו-שיח.
4. הגדר את תנאי **JOIN** של הטבלה (אם יהיו כאלה) על ידי לחיצה על לחצן **Set Table** Joins.
5. הגדר קטע **Order By** שיתבסס על שדה אחד (אם יש כזה) על ידי בחירת השדה הרצוי מתיבת הרשימה הנפתחת **Order By** ובחירה באפשרות **Asc** או **Desc**.
6. הגדר שדה **GROUP BY** יחיד על ידי בחירה באפשרות מהרשימה הנפתחת **Group By Field**.



תרשים נספח 3.12: בונה השאילתות מקל על בניית משפטי SQL

לאחר שתגדיר את הפרמטרים הדרושים, תוכל להפעיל את השאילתה, להציג את משפט SQL או להעתיק את השאילתה לחלון SQL Statement. בונה השאילתות הוא אמצעי נוח ללמידה של בניית שאילתות SELECT.

לאחר שתביא את השאילתה למבנה הרצוי (אם באמצעות Query Builder ואם על ידי הקלדה ישירה) תוכל לשמור אותה כאובייקט QueryDef במסד הנתונים. לאחר השמירה תוכל לפנות אליה מקוד Visual Basic על ידי ציון שמה. לחילופין, תוכל גם להעתיק אותה מ- Visual Data Manager ולהדביק אותה בתוכנית.

שימוש ב-Access

אם יש ברשותך עותק Access תוכל להיעזר בכלי Query Builder כדי לעצב את השאילתה תוך שימוש בממשק גרפי. לאחר מכן תוכל לפנות לשאילתה כאל אובייקט QueryDef במסד הנתונים, תוך ציון שם האובייקט בקוד.

אחד השימושים המעניינים שניתן לעשות בתוכנת Access הוא ביצוע פעולות "הנדסה הפוכה" על אובייקטי QueryDef. Access מאפשרת ליצור תיאור גרפי של הטבלאות ומסדי הנתונים שבאובייקט QueryDef מסוים, שהוזנו על ידי שימוש בשפת SQL. תהליך ההנדסה ההפוכה הוא למעשה דרך מיוחדת לבצע פעולות ניפוי שגיאות ושינוי שאילתות קיימות, תוך שימוש בממשק גרפי.

אופטימיזציית ביצועים ב-SQL

מפתחים שואפים תמיד להפיק מהיישומים ביצועיים אופטימליים בכל תחום אפשרי. גם מפתחי שאילתות SQL שואפים לאפשר לשאילתות להשיג ביצועים מירביים. כיום מוצעות מספר שיטות שמאפשרות לשפר ביצועי שאילתות.

שימוש באינדקסים

מנגנון מסד הנתונים Jet מיישם טכנולוגיית אופטימיזציה המכונה טכנולוגיית Rushmore. בתנאים מסוימים טכנולוגיה זו נעזרת באינדקסים זמינים כדי להאיץ פעולת שאילתות. אם תרצה להפיק את מירב התועלת שנובעת מיישום שיטה זו, תוכל להגדיר אינדקס לכל שדה שבשימוש קטעי WHERE או ביטויי JOIN. בעיקר כדאי לעשות זאת עבור שדות מפתח המשמשים לקישור בין טבלאות (כגון השדות Custno ו-SalesID שבמסד הנתונים לדוגמה). אינדקס גם יעבוד טוב יותר עם אופרטורים משווים מאשר עם תנאי Where, כגון LIKE או IN.

הערה:

שיטת Rushmore מסוגלת לבצע אופטימיזציה על סוגי שאילתות מסוימים בלבד. כדי שתוכל לבצע על שאילתה אופטימיזציה בשיטת Rushmore, על תנאי WHERE שבשאילתה להשתמש בשדה אינדקס. כמו כן אם תשתמש באופרטור LIKE, יש להשתמש בביטוי השוואה שיתחיל בתו רגיל ולא בתו חיפוש כללי. שיטת Rushmore ניתנת ליישום באמצעות מנגנון Jet ועל טבלאות FoxPro ו- dBase. שיטת Rushmore אינה אפשרית לשימוש עם מסדי נתונים מבוססי ODBC.

הידור שאילתות

המושג **הידור** (Compiling) שאילתה מתייחס ליצירת אובייקט QueryDef ולאחסנו במסד הנתונים. אחסון השאילתה במסד הנתונים חוסך ממתרגם הפקודות (Parser) לבנות את השאילתה מחדש בכל ריצה ומאיץ את מהירות הביצוע. לשאילתה שבשימוש נפוץ הגדר אובייקט QueryDef.

הקפדה על מבנה פשוט

בעבודה עם נתונים רבים הלקוחים מטבלאות רבות אתה עשוי להידרש לבנות משפטי SQL מורכבים מאוד. הפעלת משפטים מורכבים איטית בהרבה מזו של משפטים פשוטים. גם השימוש בקטעי WHERE המכילים מספר תנאים מעלה את רמת מורכבות השאילתה, תוך הארכת הזמן הדרוש לביצועה.

כדאי שתקפיד על מבנה פשוט עד כמה שניתן. אם יש להשתמש במשפט מורכב יחסית, ייתכן שיהיה כדאי לפצלו למספר פעולות פשוטות יותר. אם יש לבצע פעולת JOIN על שלוש טבלאות תוכל להשתמש לעיתים במשפט SELECT INTO כדי ליצור טבלה זמנית משתיים מהטבלאות, ואחר להשתמש במשפט SELECT INTO נוסף לצורך ביצוע פעולת JOIN סופית. לא קיימים כללים חד-משמעיים בנוגע למספר הטבלאות המירבי

שאליהן יכול משפט SQL להתייחס, או לגבי מספר תנאים מסוים שמעבר לו המשפט הופך מורכב. אם אתה ניתקל בבעיות בתחום הביצועים, תוכל לנסות פתרונות שונים כדי לגלות את אלה המתאימים.

דרך נוספת לפישוט משפטי SQL היא על ידי הימנעות מביצוע בדיקות התאמה לדפוסים (Pattern Matching) בקטעי WHERE. משום שבדיקות כאלו אינן מתייחסות לערכים יחידים, קשה לבצע עליהם אופטימיזציה. בנוסף, פעולות השוואה לדפוסי השוואה שהתו הראשון בהם הוא תו חיפוש כללי, איטיות יותר מפעולות להשוואה לדפוסי השוואה שהתווים הראשונים שלהם הם תווים ספציפיים. לדוגמה, אם תחפש ספרים שעוסקים בנושא SQL, איתור ספרים שהביטוי SQL מופיע במקום כלשהו בשמם (כלומר תבצע השוואה במבנה "*SQL*"= pattern) ידרוש חיפוש על כל כותר בטבלה. לעומת זאת, החיפוש אחר ספרים ששמותיהם מתחילים בביטוי SQL (כלומר השוואה לביטוי "*SQL") תאפשר לדלג על רוב הרשומות. אם הגדרת אינדקס עבור שדה הכותר, פעולת החיפוש תעבור ישירות לספר הראשון שעוסק בביטוי SQL.

העברת משפטי SQL למנגנוני מסדי נתונים אחרים

Visual Basic מסוגלת להעביר משפטי SQL דרך שרת מסד נתונים המיישם את ממשק ODBC כגון SQL Server. בהעברת משפט SQL דרך מנגנון מסד הנתונים, מנגנון Jet אינו מבצע עיבוד כלשהו של השאילתה, אלא מעביר אותה לשרת לעיבוד. חשוב שתזכור שתחביר משפטי SQL שיועבר, חייב לתאום לתחביר SQL שנתמך על ידי מסד הנתונים המארח.

כדי להשתמש ביכולות ההעברה (Pass Through) יש להגדיר את פרמטר Options של המשפט באמצעות שיטת OpenRecordset או שיטת Execute, ולהציב בו את הקבוע dbSQLPassThrough.

קובץ הפרויקט SQLDEMO.VBP שבתקליטור, מכיל רבים מבין קטעי הקוד שהוצגו בנספח זה. כל קטע קוד מקושר ללחצן. הלחיצה על לחצן יוצרת Dynaset תוך שימוש במשפט SQL שמופיע בקוד. תוצאות השאילתה מוצגות ב- Data-bound Grid. הטופס שמכיל את הפקד מכיל גם תיבת טקסט שבה יוצג משפט SQL הרלוונטי.

מכאן...

פרק זה לימד את יסודות השימוש בשפת SQL בתוכניות מסדי נתונים. למדת כיצד לבחור רשומות, וכיצד להגביל את הבחירה על ידי שימוש בקטעי WHERE. ראית גם כיצד ניתן להשתמש במשפטי SQL כדי לשנות מבנים של מסדי נתונים, וכיצד להשתמש בפונקציות צבירה להפקת תקצירי מידע.

נספח 4

דפי שרת פעילים (ASP)

מה בפרק?

- מבוא לדפי שרת פעילים (Active Server Pages)
- יצירת קבצי ASP
- גישה למסדי נתונים באמצעות ASP
- אובייקטי ASP
- ActiveX DLL בדפי ASP
- פרויקט יישום מבוסס IIS

ייתכן בשעת מעבר מדף Web אחד לאחר, תהית מהו המנגנון המתוחכם הפועל בקצה השני של הקישור. לדוגמה, כיצד חנות מקוונת מעבדת את ההזמנות שמתקבלות באמצעות אתרי Web. התשובה היא שבאתר Web פועלת תוכנית היוצרת באופן דינמי דפי Web בהם כלולות תוצאות הפעולות שלך. בעולם UNIX פעולות כאלו מבוצעות על ידי תוכניות CGI או באמצעות תוכניות הכתובות בשפת C, ואילו חברת Microsoft נכנסה לתחום על ידי שימוש בטכנולוגיית ASP - Active Server Pages (דפי שרת פעילים). טכנולוגיית Active Server Pages היא רכיב תוכנה בשרתי Web המאפשר להשתמש ב- Visual Basic לצורך כתיבת Scripts רבי עוצמה שיפעלו בשרתים. פרק זה יציג את טכנולוגיית ASP וחלק מהכלים הנתמכים על-ידה.

מבוא

אם אתה מפתח יישומים המיועדים לשימוש בסביבת Web או אינטראנט, כדאי לך לשקול שימוש בטכנולוגיית ASP. שפת HTML המשמשת ליצירת דפי World Wide Web מורכבת מתוכן סטטי. את התוכן הסטטי ניתן להעשיר על ידי שימוש בכלים כגון יישומוני Java (Java Applets), רכיבי Dynamic HTML, או פקדי ActiveX הפועלים ביישום הלקוח. בשונה מכלים אלה, טכנולוגיית ASP היא רכיב תוכנה הפועל בצד השרת, מפני שכל הקוד ב-Scripts (שרובו כתוב בשפת VBScript) פועל בשרת. השימוש ב-ASP מאפשר ליצור אתרי Web דינמיים שפועלים תוך קישור למסדי נתונים, והחזרת קוד סטנדרטי בשפת HTML הנתמך על ידי כל סוגי הדפדפנים. טכנולוגיית ASP היא פתרון נוח במצבים בהם נדרשת פנייה למשתמשי Netscape ולמשתמשי Internet Explorer, ובמצבים שבהם כדאי להימנע מבעיות הקשורות להורדת יישומונים ופקדים המיועדים לפעול ביישום הלקוח.

הערה:

שרת Web ש-Microsoft מציעה לסביבת Windows NT נקרא Internet Information Server. Microsoft מציעה גם שרת Web "פשוט" המיועד למחשבים אישיים ומכונה Personal Web Server. רכיב ASP נמצא בשני סוגי השרתים, בעת כתיבת שורות אלה ניתן להוריד את השרתים מה-Web ב- NT/95 Option Pack. בסעיף **תיקיות מדומות** תמצא פירוט קצר של ההבדלים הקיימים בין השרתים.

ASP לעומת HTML סטנדרטית

Internet Information Server מזהה דפי Web הבנויים בטכנולוגיית ASP על-פי סיומת הקובץ ASP. כדי ליצור קובץ ASP חדש, שנה סיומת קובץ HTM קיים, או צור קובץ טקסט חדש בעזרת עורך טקסט. קבצי ASP יכולים להכיל קוד Scripts או טקסט סטנדרטי של HTML, אך ברוב המקרים הם מכילים תערובת של שני סוגי הקוד. העובדה שהם מכילים קוד Scripts היא שמבדילה בין דפי ASP לבין דפי HTML רגילים.

נתייחס רגע לכתובות URL ‏(Universal Resource Locator - מציין מיקום משאבים אוניברסלי), אשר משמשות לצורך פנייה למסמכי Web. להלן שתי כתובות URL, הראשונה היא כתובת דף Web רגיל, והשנייה היא כתובת דף ASP:

```
http://www.mysite.com/mypage.htm
```

```
http://www.mysite.com/mypage.asp
```

כתובת URL רגילה (כדוגמת הכתובת הראשונה) מייצגת דף Web סטטי יחיד. כתובת דף ASP (כדוגמת הכתובת השנייה) עשויה לגרום לכך שלדפדפן המשתמש יוחזרו דפים שונים, שלכולם ניתן לפנות תוך שימוש באותה כתובת URL.

היזכר באיזה אופן מבוצעת פעולת דפדוף: פעולות כאלו מבוצעות על ידי קיום פעילות תקשורת המבוססת על העברת **בקשות** (Requests) **ותגובות** (Responses) בין הדפדפן לשרת Web. ללא תלות בפעולות המבוצעות על ידי שרת Web, התוצר הסופי של פעולות אלו הוא זרם נתוני HTML שמוחזר לדפדפן, כתגובה לבקשה שהועברה על-ידו. כאשר מתקבלת בקשה להעברת קובץ HTM, השרת נידרש רק לקרוא את הקובץ מהדיסק ולהחזיר אותו לדפדפן. אך כאשר שרת מעבד קובץ ASP הוא מבצע את כל פקודות הקוד שבו. הפקודות עשויות לגרום לשרת לבצע מיגוון פעולות רחב, ביניהן יצירת קוד HTML שיוחזר לדפדפן.

לדוגמה, נניח שתרצה ליצור דף Web שיציג את השעה העדכנית כתגובה לבקשה שתועבר אליו. השעה העדכנית היא נתון דינמי, שלא ניתן לאחסן אותו בקובץ HTML סטטי. אך ניתן לבנות דף כזה בקלות על ידי שימוש בקובץ ASP, כמו זה המוצג בתוכנית נספח 4.1.

תוכנית נספח 4.1: THETIME.ASP - קובץ ASP פשוט להצגת תאריך ושעה עדכניים.

```
<HTML>
<BODY>

The Current date and time is
<%
  Dim sTime
  sTime = Now
  Response.Write(sTime)
%>

</BODY>
</HTML>
```

קובץ ASP המוצג בתוכנית נספח 4.1 מציג את השעה והתאריך העדכניים. הקובץ מכיל תערובת של שפת HTML סטנדרטית עם קוד VBScript. אם תבצע גישה לקובץ זה כשהוא מאוחסן בשרת Web, הקוד שב-Script יבוצע בכל פעם שתלחץ על לחצן Refresh והדפדפן שלך יציג את התאריך והשעה העדכניים. אך אם משתמש ירצה לעיין בקוד המקור שבקובץ באמצעות הדפדפן, יוצג לפניו רק קוד HTML שבקובץ.

```
<HTML>
<BODY>
The Current date and time is
7/5/1998 12:30:00
</BODY>
</HTML>
```

המשתמש לא יראה את קוד VBScript המיועד לפעול בשרת, מפני שקוד זה כלל אינו מועבר לדפדפן. אך העובדה שהקוד פועל בשרת מאפשרת להשתמש בו לצורך גישה למסדי נתונים או לרכיבי תוכנה, כמתואר בתרשים נספח 4.1.



תרשים נספח 4.1: Scripts הפועלים בשרת מאפשרים ליצור אתרי Web רבי עוצמה

ספריות מדומות

בטרם תוכל להתחיל להשתמש ב-ASP תידרש לבצע מספר פעולות ניהול מערכת בשרת. קבצי האתר שלך מאוחסנים בספריה בדיסק הקשיח של שרת Web, כגון C:\InetPub\wwwroot\financeinfo\. נתיב זה מכונה **ספריה פיסית** (Physical Directory), מפני שהוא מצביע על מיקום פיסי הקיים בפועל בדיסק שלך. אך כאשר משתמש Web מעוניין לבצע גישה לקבצים המאוחסנים בספריה זו, הוא פונה אליהם על ידי שימוש בכתובת URL כגון http://myserver/finance/monthlyreport.htm. כיצד מבצע Internet Information Server את הקישור בין כתובת URL לספריה הפיסית שאותה כתובת משמשת לפנייה אליה? הקישור מתבצע בעזרת שימוש ב**ספריות מדומות** (Virtual Directories).

אם נתייחס שוב לדוגמה שלנו, שרת Web יודע שהספריה המדומה /finance היא למעשה ספריית המשנה financeinfo. ספריות מדומות הן למעשה כינויים של ספריות אמיתיות. ספריות מדומות וכלים נוספים הקשורים בפעולת שרת Web מנוהלים על ידי Internet Service Manager (בעבודה עם Personal Web Server גורמים אלה מנוהלים על ידי Personal Web Manager). בתרשים נספח 4.2 תוצגנה תמונות מסך של Internet Service Manager ושל Personal Service Manager.

יצירת ספריות מדומות לקבצי ASP

כדי ליצור ספריה מדומה בצע את הפעולות הבאות:

1. צור ספריה חדשה (לדוגמה Intpub\wwwroot\test\) על ידי שימוש בשורת הפקודה של DOS או ב- Windows Explorer.
2. פתח את Internet Service Manager, שאמור להיכלל כעת בקבוצת התוכניות בשרת שלך.

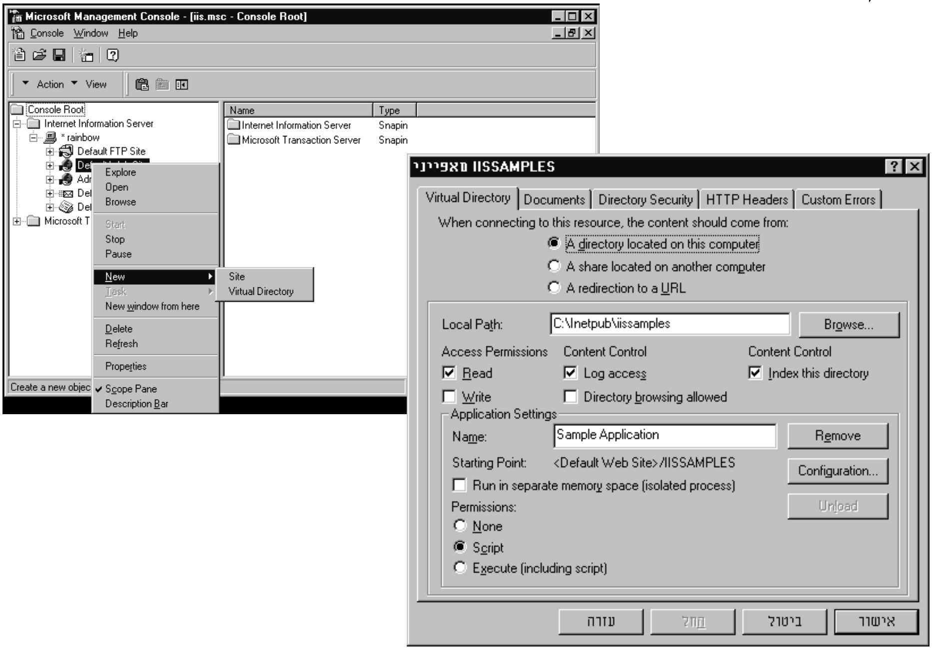
הערה:

אם יש לך הרשאות ניהול בשרת Web ניתן לנהל את IIS מרחוק.

3. התהליכים המיושמים ליצירת ספריות מדומות בגרסאות השונות של Internet Services Manager שונים מעט אלה מאלה. בתרשים נספח 4.2 מוצגות תמונות מסכים של הגרסאות השונות.
   - עבור לכרטיסיה Directories ולחץ על לחצן New.
   - אם אתה עובד ב- IIS 4.0 לחץ לחיצה ימנית על אתר Web שלך ובחר מהתפריט שיוצג את הפקודה New Virtual Directory.
   - אם אתה משתמש ב- Personal Web Manager, לחץ על הסמל Advanced ואחר כך על לחצן Add.
4. הגדר ספריה פיסית (שאותה תוכל למצוא בעזרת לחצן Browse) וספריה מדומה (שמכונה גם **כינוי** - Alias). לשם הדוגמה הגדר את test/ ככינוי לספריה החדשה.
5. הצעד הבא הוא הגדרת הרשאות גישה מתאימות עבור הספריה החדשה. סביר להניח שתוצגנה לפניך כמה אפשרויות:
   - Read Permission (הרשאת קריאה) מאפשרת למשתמשים לקרוא קבצים מהספריה המדומה. ככלל כדאי לאפשר הרשאה זו בכל הספריות.
   - Script Permission מאפשרת להפעיל Scripts המאוחסנים בספריה המדומה. כדאי שתסמן אפשרות זו כדי להשתמש בקבצי ASP בספריה.
   - Execute Permission (הרשאת ביצוע) זהה להרשאת Script Permission בתוספת אפשרות להפעיל תוכניות אחרות בספריה (ב- IIS 3.0, Script Permission אינה מוצעת כאפשרות נפרדת).
   - Write Permission מאפשרת למשתמשים להעלות קבצים לספריה.

למחיצת הדוגמה שהגדרנו יהיה עליך לאפשר Read Permission ו- Script Permission (או במקרה שאתה משתמש ב- IIS 3.0, יהיה עליך לאפשר Read Permission ו- Execute Permission).

6. לחץ על לחצן OK כדי לסגור את חלון הספריה החדשה. כעת תוכל להתחיל לאחסן קבצי ASP בספריה זו.



תרשים נספח 4.2: לגרסאות 3.0 ו-4.0 של Internet Information Server ול- Personal Web Server יש כלי ניהול שונים

אזהרה:

משום שקוד ASP פועל בשרת, כדאי שתקדיש מחשבה רבה לנושא הרשאות בספריות. אם תגדיר הרשאות Write ו-Execute לספריה מדומה, תאפשר למעשה לכל משתמש Web להעלות קבצי ASP משלו לספריה ולהפעיל אותם בשרת שלך, וזהו מצב מסוכן מאוד.

בכך סיימת את ביצוע הפעולה. כעת, לאחר שיצרת את הספריה המדומה test/, תוכל לאחסן בה קבצי ASP ולהפעיל אותם מדפדפן. בסעיף **יצירת קובץ ASP פשוט**, תיצור את קובץ ASP הראשון שלך ותשתמש בו.

יישומים והפעלות (Sessions)

בעת עיצוב אתר Web, מבנה עץ תיקיות השרת הוא בעל משמעות מבחינה ארגונית, אך בעת עבודה עם ASP יש לגורם זה משמעות רבה עוד יותר, מפני שספריות מדומות מייצגות למעשה **יישומים** (Applications). בדוגמה הקודמת, הספריה test והקבצים שאוחסנו בה נחשבו כיישום ASP. ספריות משנה שבספריות מדומות מהוות גם הן חלק מהיישום. לדוגמה, אם נעזרת ב- Windows Explorer כדי ליצור ספריית משנה בספריה test/, הספריה החדשה תהיה חלק מהיישום test/.

כאשר משתמש מבצע בפעם הראשונה גישה לדף Web כלשהו ביישום, הוא מאתחל **הפעלה** (Session) חדשה עם היישום. כל עוד המשתמש ממשיך לעבוד עם אותו יישום וממשיך בטעינת דפי Web, הוא ממשיך לעבוד באותה הפעלה. אך אם הוא עובר ליישום אחר, סוגר את הדפדפן או מפסיק את פעולתו למשך פרק זמן ארוך, שרת Web יפסיק את ההפעלה.

מדוע מונחים סמנטיים אלה חשובים: בהמשך הפרק תראה כי ASP מכילים אובייקטים שמאפשרים לשמור על מצב קיים בהפעלה. כלומר, אובייקטים אלה יאפשרו לשמור ערכים נתונים המיוחדים להפעלה מסוימת או ליישום מסוים, בשרת, ולחלוק אותם בין מספר קבצי ASP.

יצירת קבצי ASP

ליצירת קבצי ASP תוכל להיעזר בעורך טקסט כדוגמת **פנקס הרשימות** (Notepad), או בכלים מתקדמים יותר כדוגמת Visual InterDev. קובץ ASP מורכב מתערובת של קוד ותגי HTML, אך לעיתים תיתקל בקבצי ASP שיהיו על טהרת הקוד, או בקבצי ASP שיהיו על טהרת HTML. הקוד בקבצי ASP הוא קוד Script, כדוגמת הקוד שתואר בפרק 30, **שימוש ב- VBScript**.

יצירת קובץ ASP פשוט

תוכל ליצור קובץ ASP פשוט ולהתנסות בעבודה עמו, על ידי שימוש בקוד שהוצג בתוכנית נספח 4.1 ובספריה המדומה test/ שאותה יצרת קודם לכן. לשם כך יהיה עליך לבצע את הפעולות הבאות:

```
<HTML>
<BODY>

The Current date and time is
<%
 Dim sTime
 sTime = Now
 Response.Write(sTime)
%>

</BODY>
</HTML>
```

1. קרא לקובץ THETIME.ASP ושמור אותו בספריה test/.
2. פתח את הקובץ בדפדפן, באמצעות הכתובת http://myserver/test/thetime.asp, במקום myserver הצב את שם השרת שלך.

אם הקוד יוצג על המסך במקום לרוץ, ודא שסיומת שם הקובץ היא ASP ושהגדרת את ההרשאות הדרושות לספריה המדומה.

הערה:

שרתי Microsoft Web מאפשרים להגדיר שמות ברירת מחדל לקבצים (לרוב DEFAULT.ASP או DEFAULT.HTM). יכולת זו מבטלת את הדרישה להקליד כתובת URL מפורטת עד לרמת הקובץ. בהקלדת http://myserver/test, שם הקובץ DEFAULT.ASP או DEFAULT.HTM יוצג אוטומטית, אם קובץ זה קיים בספריה test/.

תגי Script צד השרת

בעת עיבוד קובץ ASP, השרת מבצע רק את הקוד שעוצב כ-Script צד השרת (Server Side Script), שאר חלקי הקובץ נשלחים חזרה לדפדפן, כפי שהם. את קוד צד השרת שבקבצי ASP ניתן לסמן בכמה אופנים:

- שימוש בתג <SCRIPT> HTML עם האפשרות RUNAT=SERVER.
- שימוש בסימני <% וכן %> לסימון תחילה וסוף Script צד השרת. תחביר זה קצר יותר ואני ממליץ להשתמש בו.

בכל שיטה בה תסמן את ה-Script, ודא שהוא תחום במפרידים, כמו בדוגמה הבאה:

```
<SCRIPT LANGUAGE="VBSCRIPT" RUNAT="SERVER">
Dim I
For I = 1 to 10
  Response.Write ("This is Line " & I & "<BR>")
Next
</SCRIPT>
```

תוכל להשתמש גם בשיטת הסימון הקצרה יותר:

```
<%
Dim I
For I = 1 to 10
  Response.Write ("This is Line " & I & "<BR>")
Next
%>
```

שים לב שבעת שימוש בשיטה השנייה לא תידרש להגדיר את שפת ה-Script. זאת היות ו-VBScript היא שפת ברירת מחדל לכתיבת קבצי ASP. אך יש לזכור ששפת ברירת המחדל היא פרמטר שניתן להגדירו בהגדרת הפרמטרים של השרת, כך שייתכן שתרצה להציב את השורה הבאה בתחילת כל קובץ ASP:

```
<%@ LANGUAGE="VBSCRIPT" %>
```

שורת הוראה זו מורה למנגנון Scripting (Scripting Engine), להגדיר את VBScript כשפת ברירת המחדל של הדף המעובד כעת.

ניתן למקם תגי Script בכל מקום בקובץ, למעט מספר הגבלות:

- יש מספר הוראות (Directives) ואפשרויות, כגון שורת הקוד הקודמת והאפשרות Option Explicit שנדרשות להופיע בתחילת הקובץ, עוד לפני תג <HTML>.
- אם הדף שהגדרת מפנה את המשתמש לאתר אחר, לא תוכל להעביר תגי HTML חזרה לדפדפן המשתמש (בהמשך הפרק תובא הדגמה לאפשרות זו).
- לא ניתן לשלב תגי HTML "נקיי" בקטע התחום בין תגי <SCRIPT> לא אם כן משתמשים בפקודה Response.Write.

חשוב גם שתזכור שקוד VBScript אינו בעל יכולות מלאות כמו קוד Visual Basic, יש מספר הבדלים ומגבלות כמתואר בפרק 30 **שימוש ב-VBScript**.

ראה, "מבוא ל-VBScript" ו"שפת VBScript" פרק 30.

דפי Web פשוטים אך דינמיים

כעת, לאחר שלמדת מעט על יסודות טכנולוגיית ASP, נכתוב קובץ ASP נוסף. במשרדי יש מחשב ששמו cdtower, משמו ניתן להבין שהוא מקושר ל"מגדל" כונני תקליטורים. מחשב זה נועד לאפשר למשתמשים ברשת שלנו להתקשר לכונן תקליטורים לצורך התקנת תוכנות. הקצב הגבוה שבו מוחלפים התקליטורים בכוננים אלה מקשה מאוד לדעת ברגע נתון איזה תקליטור מצוי באיזה כונן. הייתי רוצה שתהיה לי אפשרות לנהל את המעקב אחר הנושא באופן ממוכן. למרבה המזל השימוש ב- Personal Web Server ו-ASP מקל על ביצוע פעולות מעקב אלו.

עליך להשתמש לשם כך באובייקט FileSystemObject כדי לזהות את **תווית הדיסק/תקליטור** (Volume Label) של אמצעי האחסון המצוי בכל כונן שמקושר למחשב. השימוש בקטע קוד זה בקובץ ASP שפועל בשרת יאפשר להציג את תוויות התקליטורים המצויים בכוננים המקושרים לשרת בדפדפן מרוחק. קטע הקוד בתוכנית נספח 4.2 הוא פשוט מאוד להבנה.

תוכנית נספח 4.2: VOLLABEL.ASP

```
<HTML>
<BODY>
<H1>Drive List</H1><HR>
<%
  Dim objFileSys
  Dim objDrives
  Dim objDrive

      'IGNORE DISK NOT READY ERRORS
      On Error Resume Next

  'CREATE REFERENCES TO FILE SYSTEM AND DRIVES COLLECTION
```

```
   Set objFileSys = Server.CreateObject("Scripting.FileSystemObject")
   Set objDrives = objFileSys.Drives

   'DISPLAY VOLUME LABELS
   For Each objDrive in objDrives
     if objDrive.DriveLetter > "C" Then
       Response.Write ("The CD in Drive " & objDrive.DriveLetter)
       Response.Write (" is " & objDrive.VolumeName & ". <BR> ")
      End If
   Next

   'CLEAN UP!
   Set objDrives = Nothing
   Set objFileSys = Nothing
%>
</BODY>
</HTML>
```

כאשר משתמש יבצע גישה לדף זה, קטע הקוד יתבצע בשרת. דף Web שהובא בתוכנית נספח 4.2 מבצע את פעולת המעקב, אך הוא מותיר מקום רב לשיפורים במה שקשור לעיצוב הפלט שלו. תוכל לשפר את חזות הקלט על ידי שימוש בתגי HTML כגון תג <TABLE>. תוכל להשתמש בתגי אלה על ידי הכנסת שינויים בפקודות Response.Write שבקובץ (כפי שבוודאי כבר הבנת, פקודת Response.Write משמשת להעברת פלט HTML חזרה לדפדפן).

בדוגמה הצגת את הערך במאפיין VolumeName של אובייקט Drive, אך בוודאי תרצה להציג תיאור קצר של התקליטור. תוכל לבצע זאת על ידי הגדרת פונקציה מותאמת אישית בשם sDescription, שתחזיר את תיאור אמצעי האחסון שלו:

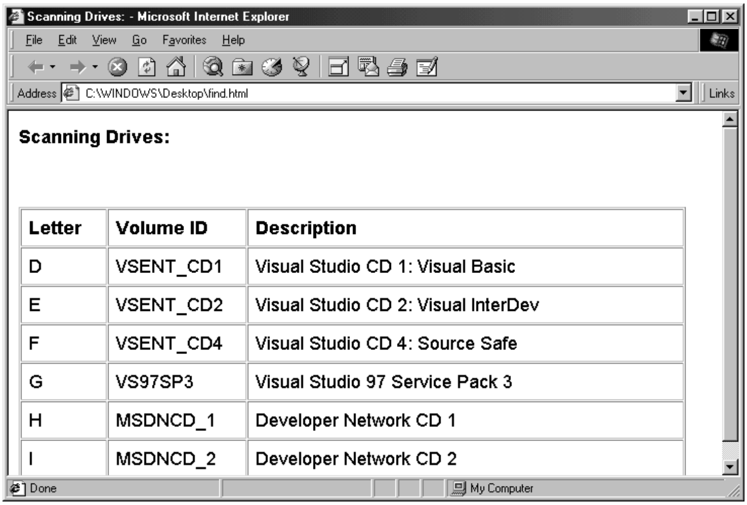
```
Function sDescription(sVolName)
 Select Case UCase(sVolName)
   Case "BACKUP0398"
    sDescription = "Backup of critical files 3/98"
   Case "DN_60ENUD2"
    sDescription = "Developer's Network CD #2"
   '
   '' Add More Descriptions here
   '
   Case Else
   sDescription = "Unknown"
 End Select
End Function
```

תוכל לשלב פונקציות ושגרות משלך בקבצי ASP, כפי עשית זאת ב- Visual Basic. הוסף את הפונקציה sDsecription לקובץ על ידי הקלדת הקוד בתחילת הקטע המכיל את קוד

VBScript. הפונקציה תהיה מוכרת בכל הדף בו ייכלל הקוד שלה, ותוכל לקרוא לה באותו אופן בו תקרא לפונקציה רגילה של Visual Basic :

```
Response.Write sDescription(objDrive.VolumeName)
```

בתרשים נספח 4.3 ניתן לראות את המבנה הסופי של דף ASP, הכולל את הטבלה שבה מוצגים התיאורים.



תרשים נספח 4.3: קובץ ASP שבו מוצגים נתוני תקליטורים

קבצים כלולים

כאשר תכתוב קוד VBScript ב-ASP , תוכל לכתוב את הפונקציות והשגרות בתחילת הקובץ. ולבצע אליהן קריאות מכל מקום בדף. כך תוכל למשל להגדיר פונקציה שתקלוט מספר ותיצור קובץ HTML תוך שימוש במספר שנקלט :

```
Sub DisplayAmount(nAmount)
  Dim sHTML
   sHTML = "<FONT "
   If nAmount < 2000 Then sHTML = sHTML & "COLOR=RED"
   If nAmount < 4000 Then sHTML = sHTML &
               "COLOR=GREEN"
  sHTML = sHTML & ">"
  sHTML = sHTML & FormatCurrency(nAmount)
  sHTML = sHTML & "</FONT>"
   Response.Write sHTML
End Sub
```

השיגרה DisplayAmount קולטת ערך מספרי ואחר מגדירה את מבנה התצוגה ואת צבע הרקע על-פי הערך שנקלט. פרוצדורות כאלו עשויות להיות שימושיות מאוד בקבצי

ASP שונים, אך לא קיימת דרך המאפשרת להגדיר שיגרה כציבורית כך שתהיה נגישה לכלל קבצי ASP. תוכל לצרף אותה לקבצים שונים באמצעות חיתוך והדבקה, אך מצב כזה יקשה על תחזוקת העותקים השונים. את הבעיה תוכל לפתור על ידי העברת השיגרה לקובץ נפרד והוספת הקובץ לכל דף ASP בו תידרש השיגרה. להוספת קבצים אחרים לקובץ ASP הפעיל, יש להשתמש בפקודת INCLUDE#, כמו בשתי הדוגמאות הבאות:

```
<!-- #INCLUDE VIRTUAL = "/test/formatfuncs.asp" --→
```

```
<!-- #INCLUDE FILE="C:\InetPub\wwwroot\test\formatfuncs.asp" --→
```

הדוגמה הראשונה משתמשת בתיקיה וירטואלית לצורך פנייה לקובץ ואילו השנייה משתמשת בתיקיה פיסית. התוצאה המושגת בשני האופנים היא הוספת הקוד שבקובץ formatfuncs.asp לקובץ ASP הפעיל בטרם ביצוע הקוד שבקובץ ASP. אחד השימושים הנפוצים של פקודת INCLUDE# הוא לצורך הוספת כותרות תחתונות המכילות מידע כגון כתובות וכתובות E-Mail לדפי Web.

גישה למסדי נתונים באמצעות דפי שרת פעילים

נניח שתרצה לפרסם תוכן מסד נתונים כלשהו ב-Web. ASP יאפשר לעשות זאת באמצעות שילוב אובייקטי ADO בקוד, שיבצעו את פעולות הגישה שתוארו בפרק 28, **גישה לאובייקטי נתונים באמצעות פקדי ActiveX**. כדי שתוכל לבצע גישה למסד הנתונים, יש להגדיר ***מקור נתונים*** (Data Source). מסד הנתונים הפיסי יוכל להימצא בשרת או בכל מחשב אחר, כל עוד תוכל להתקשר אליו תוך שימוש במקור הנתונים.

בעת הגדרת מקורות נתונים שיפעלו בשילוב עם ASP, כדאי להגדיר System DSN ולא User DSN. השימוש ב- System DSN הופך את הנתונים לזמינים לגישה בכל מצב, ולא רק כאשר משתמש מסוים פעיל במערכת.

ראה, "הגדרת מקור נתונים", פרק 28.

הערה:

ניתן גם להגדיר חיבור שאינו מבוסס על DSN (DSN-less Connection) שבו אין צורך להגדיר מקור נתונים. גם שיטת עבודה זו תוארה בפרק 28.

הדוגמאות המובאות בפרק זה מתבססות על ההנחה שהגדרת מסד הנתונים BIBLIO כמקור נתונים, כפי שתואר בפרק 26, **שימוש באובייקטי גישה לנתונים (DAO)**.

ראה, "הגדרת פרויקט הכולל אובייקטי DAO", פרק 26.

הערה:

אם בעת עבודה עם שרת Web תיתקל בבעיות התחברות למקור נתונים מבוסס SQL Server שפועל על מחשב אחר, שעבורו מוגדרות הגדרות אבטחה רגילות, אתה עשוי להיתקל בהודעת שגיאה כגון: Connection Open error in the function CreateFile(). הודעת שגיאה זו מציינת שלחשבון שממנו מתבצע החיבור (בדרך כלל חשבון I_USER) אין הרשאות מספיקות ב- SQL Server.

הפעלת שאילתה על מסד נתונים

הידע הדרוש להפעלת שאילתת ADO והצגת התוצאות בדף Web כבר מצוי בידך. אך אפילו מסד נתונים קטן יחסית, כגון BIBLIO.MDB מכיל רשומות רבות. לכן יש להגדיר את השאילתה כך שתוחזרנה רק הרשומות הרצויות למשתמש, ומכאן משתמע שעל התוכנית לטפל בקלט מהמשתמש המגדיר את הרשומות.

יצירה של דוגמת דף שאילתה

תג <FORM> המובנה ב-HTML מאפשר לשלוח לשרת קלט שהתקבל מהמשתמש. כעת ניצור דף HTML שיכיל טופס, בשם DBQUERY.HTM, שיאפשר למשתמש להזין שם מחבר, ולבצע חיפוש אחר רשומות המכילות את השם. להגדרת הטופס יש להזין את הקוד בתוכנית נספח 4.3 ולשמור אותו בשרת.

תוכנית נספח 4.3: הגדרת טופס HTML בשם DBQUERY.HTM

```
<HTML>
<BODY>
<H1>Biblio Database Search</H1><HR>

<FORM ACTION=dbsearch.asp METHOD=POST>

Enter Author to search for:
<INPUT TYPE=TEXT NAME=txtSearch>
<INPUT TYPE=SUBMIT VALUE="Begin Search">

</FORM>
</BODY>
</HTML>
```

הערה:

טופס HTML מורכב מתגי <INPUT> מסוגים שונים, אשר מוצגים בדפדפן בצורת שדות להזנת נתונים. תוכן השדות מועבר לשרת כאשר המשתמש "שולח" את הטופס לשרת.

הקוד בתוכנית נספח 4.3 אינו קוד ASP. בקוד אין פקודות VBScript והוא מאוחסן בקובץ עם סיומת HTM. קטע הקוד משמש רק להעברת טופס HTML לדף ASP בשם DBSEARCH.ASP. הדף DBSEARCH.ASP, שהקוד בו מובא בתוכנית נספח 4.4, מבצע את החיפוש הממשי.

גם הקוד ב-DBSEARCH.ASP הוא פשוט יחסית והוא מבצע שלוש פעולות :

- מאחזר את הערך הרצוי למשתמש מהשדה txtSearch שבטופס ומשתמש בו לבניית שאילתה.
- מפעיל את השאילתה על ידי שימוש ב-ADO לקבלת מערך רשומות.
- שולח את תוכן מערך הרשומות חזרה לדפדפן.

בתוכנית נספח 4.4 תוכל לראות שהשאילתה בקובץ DBSEARCH.ASP בנויה על ביטוי Like.

תוכנית נספח 4.4: DBSEARCH.ASP ביצוע חיפוש במסד נתונים

```
<HTML><BODY>
<%
	Dim cn
	Dim rs
	Dim sSQL
	Dim sSearchString

	'GET THE SEARCH STRING FROM THE FORM
	sSearchString = Request.Form("txtSearch")

	If sSearchString = "" Then
		Response.Write ("No search string entered!")
		Response.End
	End If

	'CONNECT TO THE DATABASE AND PERFORM THE SEARCH
	Set cn = Server.CreateObject("ADODB.Connection")
	cn.Open "DSN=BIBLIO"

	sSQL = "SELECT * FROM AUTHORS WHERE AUTHOR LIKE "
	sSQL = sSQL & "'" & sSearchString & "%' ORDER BY AUTHOR"

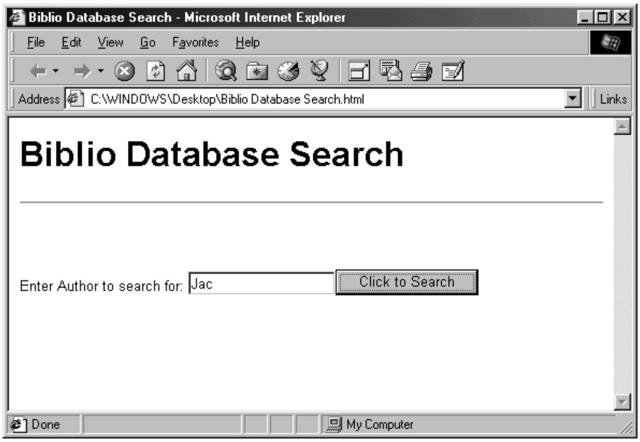
	Set rs = cn.Execute(sSQL)

%>
<TABLE BORDER=1>
<TR>Author's Name</TR>
<%

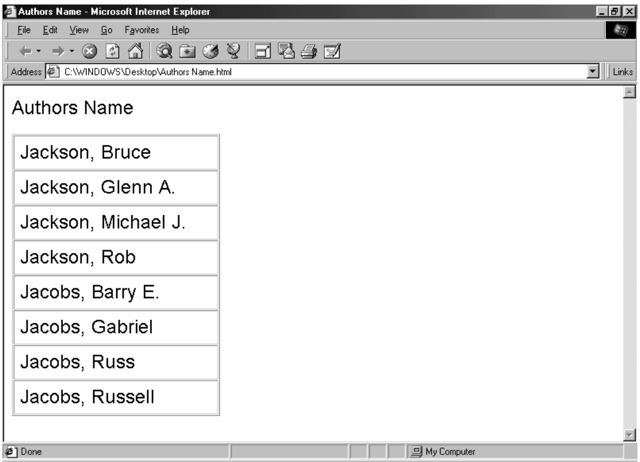
	'DISPLAY THE RESULTS IN A TABLE
	Do While Not rs.EOF
		Response.Write ("<TR><TD>")
		Response.Write rs.Fields("Author")
		Response.Write ("<TD></TR>")
	rs.MoveNext
	Loop
```

```
        rs.Close
        cn.Close
        Set rs = Nothing
        Set cn = Nothing
%>
</TABLE>
</BODY>
</HTML>
```

לאחר שתיצור את הקבצים DBQUERY.HTM ו-DBSEARCH.ASP, תוכל לפתוח את כתובת URL של הקובץ DBQUERY.HTM ולהקליד כמה אותיות וכך תוצגנה הרשומות התואמות למחרוזת שהקלדת (ראה תרשימים 4.4 ו-4.5).



תרשים נספח 4.4: דף Web המוצג כאן מכיל טופס HTML סטנדרטי, שמאפשר למשתמש להזין ערך ולהעביר אותו לשרת לעיבוד



תרשים נספח 4.5: קוד VBScript שפועל בשרת מחזיר את תוכן המערך כטבלת HTML

הערה:

בדוגמת החיפוש במסד נתונים השתמשנו בקבצי ASP ו-HTM נפרדים. באותה מידה יכולנו לשלב את שני הקבצים בקובץ ASP אחד. השינוי היחיד שיש לבצע הוא שינוי ערך הפרמטר action של הטופס כך שהוא יפנה לאותו דף. ייתכן שתרצה גם להיעזר בביטוי If כדי לבדוק האם יש תוצאות שיש להציגן:

```
<% If Request.Form("txtSearch") <> "" Then DisplayResults %>
```

דוגמת הקוד שהוצגה כאן מבוססת על ההנחה שכתבת פונקציה מותאמת אישית בשם DisplayResults שמפעילה את השאילתה.

הצגת נתונים בעזרת קישורים

הקוד בתוכניות נספח 4.3 ו-4.4 משתמש בשדה בטופס HTML לקבלת קריטריון החיפוש מהמשתמש. על המשתמש להקליד את קריטריון החיפוש בתיבת טקסט ואחר ללחוץ על לחצן. במקרים רבים כדאי להיעזר ב**קישורים** (Hyperlinks) לביצוע פעולות ניווט במסד הנתונים. לדוגמה, כדי לאפשר למשתמש לבחור שם מחבר מתוך רשימה ולהציג את ספריו. תוכל לעשות זאת בקלות על ידי הגדרת Hyperlink לכל מחבר. כדי ליצור את הקישור שנה את הקוד בלולאת While שבתוכנית נספח 4.4 באופן הבא:

```
'DISPLAY THE RESULTS IN A TABLE
	While Not rs.EOF
		Response.Write ("<TR><TD><A HREF=author.asp?id=")
		Response.Write rs.Fields("AU_ID") & ">"
		Response.Write rs.Fields("Author")
		Response.Write ("</A><TD></TR>")
			rs.MoveNext
	Wend
```

בתוצאות החיפוש, יוצג שם סופר בצורת קישור לקובץ ASP חדש, AUTHOR.ASP. אם תשתמש בפקודה View Source של הדפדפן, תוכל לראות שכל קישור הוא ייחודי:

```
<A HREF=author.asp?id=16061>Jackson, Bruce</A>
```

שורת HTML זו מעבירה את הפרמטר id לדף ASP, AUTHOR.ASP. אם תלחץ על הקישור, הדפדפן יבקש מהשרת את כתובת URL הבאה:

```
http://bshome/test/author.asp?id=16061
```

שים לב שכתובת URL מכילה פרמטר id, שמופרד מכתובת URL הבסיסית על ידי סימן שאלה. אם יהיו לשאילתה פרמטרים נוספים, הם יופרדו זה מזה בתווי &. תחביר השימוש בפרמטרים בכתובות URL הוא כדלקמן:

```
http://targetURL? parm1name= parm1value & parm2name=parm2value &
parm3name=parm3value...
```

אוסף הפרמטרים בכתובת URL מכונה מחרוזת שאילתה (Query String), וניתן לאחזר את הפריטים שבו על ידי שימוש באוסף QueryString של אובייקט Request.

```
If Request.QueryString("id") = "" Then Response.Write "No ID Entered"
```

ניתן לאחזר ערך במחרוזת שאילתה שהועברה לשרת, ולהשתמש בו בשאילתה לאחזור נתונים ממסד נתונים, כפי שנעשה בדוגמת שימוש בשיטה Request.Form בתוכנית נספח 4.4.

```
sSQL = "SELECT Title From [TITLE Author], [Titles] "
sSQL = sSQL & " WHERE [Title Author].ISBN = [Titles].ISBN "
sSQL = sSQL & " and [Title Author].AU_ID=" & Request.QueryString( "id")
```

כתרגיל, צור כעת את הדף AUTHOR.ASP תוך שימוש בשאילתה שלעיל. מבנה קובץ ASP יהיה זהה לזה שבתוכנית נספח 4.4, פרט להבדלים בשמות השדות.

עדכון נתונים במסד נתונים

השימוש ב-ASP להצגת נתונים הוא מועיל מאוד, אך בשלב כלשהו יש לבצע פעולות עריכה. להלן שני אופנים שבהם תוכל להעביר קלט מדפדפן המשתמש לדף ASP:

- משלוח נתונים (Posting) על ידי שימוש בקובץ HTML.
- הוספת פרמטרים למחרוזת שאילתה.

שתי השיטות הן שיטות מועילות וכל אחת מהן מתאימה לשימוש במצבים מסוימים, אך משלוח נתונים מקנה גמישות רבה יותר בעבודה עם טפסי HTML. בדוגמאות שהצגנו עד כה עבדנו עם שדות קלט מסוג Text בלבד. אך HTML תומכת באמצעים נוספים להעברת קלט כגון לחצני אפשרויות, תיבות רשימה נפתחת, ואזורי טקסט להזנה חופשית (Free-form text Areas). סוג השדה HIDDEN (נסתר) מועיל מאוד בפעולות עריכה במסדי נתונים. שדה מסוג זה דומה לשדה רגיל מסוג טקסט, אך המשתמש אינו יכול לראות או לשנות את הערך המאוחסן בו. בעת עבודה עם קבצי ASP ניתן ליצור שדות מסוג HIDDEN ולהעבירם לדפדפן. כאשר המשתמש יעביר את הטופס חזרה לשרת, תוכן שדות אלה יועבר לשרת יחד עם השדות האחרים.

כעת נחזור לקבצים שהוצגו בתוכניות נספח 4.3 ו-4.4 ונוסיף להם יכולת לעריכת שם המחבר ושנת הלידה שלו. בנוסף, נאחד את כל הפונקציות שמופיעות בדוגמאות הקודמות בדף ASP יחיד:

- הפונקציה AskForAuthors תחליף את קובץ DBQUERY.HTM.
- הפונקציה GetAuthorList תחליף את קובץ DBSEARCH.ASP.
- DisplayEditScreen היא שיגרה חדשה, שתציג טופס לעריכת נתוני המחבר.
- הפונקציה UpdateDBInfo תשנה את מסד הנתונים.

הקוד לדף ASP החדש AUTHOREDIT.ASP מובא בתוכנית נספח 4.5.

תוכנית נוספת 4.5: AUTHOREDIT.ASP ביצוע פעולות חיפוש ועריכה.

```
<HTML>
<BODY>
<%
Dim cn
Dim rs
Dim sSQL
Const MYASPNAME="authoredit.asp"

Sub AskForAuthors()

    Response.Write ("<H1>Biblio Database Search</H1><HR>")
    Response.Write ("<FORM ACTION=" & MYASPNAME & "?mode=search METHOD=POST>")
    Response.Write ("Enter Author to search for:")
    Response.Write ("<INPUT TYPE=TEXT NAME=txtSearch>")
    Response.Write ("<INPUT TYPE=SUBMIT VALUE=""Click to Search"">")
    Response.Write ("</FORM>")

End Sub

Sub GetAuthorList()

    'CONNECT TO THE DATABASE AND PERFORM THE SEARCH
    Set cn = Server.CreateObject("ADODB.Connection")
    cn.Open "DSN=BIBLIO"

    sSQL = "SELECT * FROM AUTHORS WHERE AUTHOR LIKE "
    sSQL = sSQL & "'" & Request.Form("txtSearch")
    sSQL = sSQL & "%' ORDER BY AUTHOR"
    Set rs = cn.Execute(sSQL)

    'DISPLAY THE RESULTS IN A TABLE
    Response.Write("<TABLE BORDER=1><TR>Author's Name</TR>")
    Do While Not rs.EOF
        Response.Write ("<TR><TD><A HREF=" & MYASPNAME & "?id=")
        Response.Write rs.fields("AU_ID") & "&mode=dispedit>"
        Response.Write rs.Fields("Author")
        Response.Write ("</A><TD></TR>")

      rs.MoveNext
    Loop

    rs.Close
    cn.Close
End Sub
```

```
Sub DisplayEditScreen()

    'CONNECT TO THE DATABASE AND GET THIS AUTHOR'S INFO
    Set cn = Server.CreateObject("ADODB.Connection")
    cn.Open "DSN=BIBLIO"

    sSQL = "SELECT * FROM AUTHORS WHERE AU_ID= " & Request.QueryString("id")
    Set rs = cn.Execute(sSQL)

    'GENERATE THE HTML FORM
    Response.Write ("<H1>Edit Author Information</H1><HR>")
    Response.Write ("<FORM ACTION=" & MYASPNAME & "?mode=updatedata METHOD=POST>")
    Response.Write ("Name:<INPUT TYPE=TEXT NAME=txtName")
    Response.Write (" VALUE='" & rs.Fields("Author") & "'><BR>")
    Response.Write ("Year Born:<INPUT TYPE=TEXT NAME=txtYear")
    Response.Write (" VALUE='" & rs.Fields("Year Born") & "'><BR>")
    Response.Write ("<INPUT TYPE=HIDDEN NAME=AuthorID VALUE=")
    Response.Write (rs.Fields("AU_ID") & ">")

    Response.Write ("<INPUT TYPE=SUBMIT VALUE=""Update Info"">")
    Response.Write ("</FORM>")

End Sub

Sub UpdateDBInfo()

    'CONNECT TO THE DATABASE
    Set cn = Server.CreateObject("ADODB.Connection")
    cn.Open "DSN=BIBLIO"

    'BUILD SQL UPDATE STATEMENT
    sSQL = "UPDATE AUTHORS SET Author='" & Request.Form("txtName") & "',"
    sSQL = sSQL & "[Year Born]=" & Request.Form("txtYear")
    sSQL = sSQL & " WHERE AU_ID=" & Request.Form("AuthorID")
    cn.Execute(sSQL)

    'DISPLAY A MESSAGE
    Response.Write ("<H1> Information Updated!</H1>")
    AskForAuthors
End Sub

Select Case Request.QueryString("mode")
    Case "search"
       GetAuthorList
    Case "dispedit"
       DisplayEditScreen
    Case "updatedata"
       UpdateDBInfo
```

```
    Case Else
        AskForAuthors
End Select

Set rs = Nothing
Set cn = Nothing
%>
</TABLE>
</BODY></HTML>
```

הערה:

בעת עבודה עם קלט מהמשתמש, יש להקדיש תשומת לב מיוחדת לגרשיים. במקרים מסוימים, לפני שתוכל להשתמש בערכי שדה במשפטי SQL, יש לכתוב פונקציה שתטפל בתווי אלה.

בתוכנית נספח 4.5 מובא קובץ ASP אך הקובץ מאפשר ליצור ארבעה מסכי HTML נפרדים. הפרמטר mode שבמחרוזת השאילתה קובע את הפעולה שקובץ ASP יבצע.

אובייקטי ASP

לדפי ASP מספר סוגי אובייקטים, שניתן לגשת אליהם מקוד Scripts. בפרק זה הצגנו כבר מספר דוגמאות לשימוש באובייקטים מסוג Response (תגובה) ו-Server (שרת). בסעיף זה נחקור את השימוש בסוגי האובייקטים אלה ביותר פירוט ונציג דוגמאות לשימוש בהם. האובייקטים שבהם תומכת ASP הם:

- Session (הפעלה)
- Response (תגובה)
- Request (בקשה)
- Server (שרת)
- Application (יישום)

ניהול אבטחה בעזרת אובייקט Session

קודם לכן כבר ציינו כי IIS כולל יכולות מסוימות שמאפשרות ליישומי ASP לשמור נתונים מסוימים במעבר בין דפים. תוכל למשל להציג טופס שבו המשתמש יתבקש לבחור מדינה מתוך רשימה, ויעביר את הערך שנבחר לדף ASP. דף ASP יוכל לאחסן את הערך בשרת, באופן שיאפשר לדפים אחרים גישה אליו. פעולות כאלו מבוצעות על ידי שימוש ב**משתני הפעלה** (Session Variables). ההצהרה על משתני ההפעלה והגישה אליהם מתבצעות כמו הפעולות המקבילות באוסף. כדי ליצור משתנה הפעלה יש להגדיר שם וערך בלבד:

```
Session("Country")="United States"
```

אחד השימושים השכיחים ביותר במשתני הפעלה הוא בפעולות אבטחה. כאשר אתה מאחסן מסד נתונים ב-Web, סביר להניח שאינך מעוניין שכל משתמש אקראי יוכל לבצע בו שינויים. ייתכן אף שתרצה להגביל את הגישה באופן שיאפשר רק לאנשים מסוימים להציג נתונים מסוימים. הפעולה הטבעית ביותר להוספת אבטחה לאתר, תהיה הצגת דף כניסה (Login Page) למערכת, ורק משתמשים מורשים יוכלו לעבור אותו ולהיכנס לאתר. דף הכניסה יוכל להיות דף HTML פשוט, ובו שדות להזנת שם משתמש וסיסמה. הטופס יעביר את הנתונים שיוזנו לקובץ ASP, שיאמת את השם והסיסמה.

אך גם אם תוסיף לאתר דף כניסה כזה, כיצד תמנע ממשתמשים לעקוף אותו על-ידי הזנת כתובת? תוכל לעשות זאת על ידי שימוש במשתני הפעלה. נניח שיש לך קובץ ASP בשם REPORT.ASP שאותו אתה מעוניין לאבטח. תוכל להוסיף בראש הדף בדיקת If פשוטה שתוודא אם במשתנה הפעלה מסוים הוצב הערך הרצוי:

```
<%
	If Session("UserName")="" Then
		Response.Write "You are not logged in!<BR>"
		Response.End
	End If
	Response.Write "Welcome, " & Session("UserName")
%>
```

פקודת If תבדוק תוכן משתנה הפעלה בשם UserName ואם המשתנה ריק, היא תסיים את העברת התגובה מהשרת. כלומר, התוכן בקובץ REPORT.ASP שמוצג לאחר פקודת If, יוצג לפני משתמשים שהציבו ערך כלשהו במשתנה ההפעלה UserName בלבד.

טיפ:

כדי לאבטח מספר דפי Web, כדאי שתבצע את פעולת האבטחה על ידי שימוש בקובץ Include. אופן פעולה כזה יקל עליך את תחזוקת תוכנית האבטחה.

לאימות שמות המשתמשים והסיסמאות ולהצבת ערך במשתנה ההפעלה UserName יש להיעזר בקוד ASP. קטע הקוד הרלוונטי, המוצג בתוכנית נספח 4.6, יאמת את הנתונים שיתקבלו מטופס הכניסה, ואם המשתמש הזין סיסמה תקינה, הוא יציב ערך במשתנה UserName.

תוכנית נספח 4.6: CHECKUSER.ASP - אימות זהות משתמשים מול מסד נתונים

```
<%
    Dim cn
    Dim rs
    Dim sCheckName
    Dim sCheckPW

    'CLEAR SESSION VARIABLE
    Session("UserName")=""

    'GET USERNAME AND PASSWORD FROM THE FORM
    sCheckName = Request.Form("txtusername")
    sCheckPW = Request.Form("txtpassword")

    'IF THEY DIDN'T ENTER ANYTHING THEN RETURN TO LOGIN PAGE
    If sCheckName = "" then Response.Redirect "login.html"
    'CHECK USER PASSWORD IN THE DATABASE
    Set cn = Server.CreateObject("ADODB.Connection")
       cn.Open "Driver=Microsoft Access Driver (*.mdb);DBQ=C:\data\security.mdb"
    sSQL = "SELECT * FROM UserList WHERE UserName='" & sCheckName & "'"
    Set rs = cn.Execute(sSQL)

    'IF THEY AREN'T IN DATABASE THEN RETURN TO LOGIN PAGE
    If rs.EOF Then
        rs.Close
        cn.Close
        Response.Redirect "login.html"
    End If

    'IF PASSWORD DOESN'T MATCH THEN RETURN TO LOGIN PAGE
    If UCase(rs.Fields("Password")) <> UCase(sCheckPW) Then
        rs.Close
        cn.Close
        Response.Redirect "login.html"
    End If

    rs.Close
    cn.Close

    'PASSWORD IS VALID! - SET SESSION VARIABLE
    Session("UserName")=sCheckName

    'GO ON TO REPORT PAGE
    response.redirect "report.asp"
%>
```

קטע הקוד בתוכנית נספח 4.6 משתמש בפקודה Response.Redirect שמפנה את דפדפן המשתמש לכתובת URL אחרת. אם יוזן שילוב לא חוקי של שם וסיסמה, המשתמש יוחזר לדף הכניסה. לפעמים תרצה לשלב בדף מונה שיעביר התראה למנהל המערכת, אם משתמש יבצע מספר נסיונות כניסה שעובר מספר שקבעת.

אזהרה:

לפעולות האבטחה שתוארו ניתן להתייחס כפעולות **אבטחה ברמת היישום** (Application Level Security). פעולות אלו אינן מספקות **אבטחה ברמת הרשת** (Network-level Security). במצב כזה, כדי לזהות סיסמאות המועברות לשרת משתמשים יכולים להשתמש ברכיבי חומרה מיוחדים להאזנה לתמסורת המועברת לשרת Web וממנו, כמו בהאזנה לקווי טלפון. כדי למנוע זאת, שקול שימוש בפרוטוקול https.

משום שהדפדפן פועל על בסיס העברת בקשות ותגובות, המשתמש אינו נדרש לקיים תקשורת רציפה עם שרת Web. לכן לא תמיד השרת יכול לזהות מצבי ניתוק תקשורת, כך שאם חלף משך זמן מסוים מאז סיום הפעולה האחרונה, כדאי להפסיק את ההפעלה. את משך הזמן תוכל להגדיר במאפיין TimeOut של אובייקט Session :

```
Session.TimeOut = 60
```

שורת הקוד תציב את הערך 60 (בדקות) במאפיין TimeOut של אובייקט Session. הצבת הערך תגרום לכך שאם תחלופנה 60 דקות ללא פעולה, השרת יסיים את ההפעלה המקשרת אותו למשתמש, תוך השמדת משתני ההפעלה השייכים להפעלה. תוכל גם לסיים הפעלה באופן מכוון על ידי השיטה Abandon, כמו בדוגמה הבאה :

```
Session.Abandon
```

שליטה בפלט באמצעות אובייקט Response

האובייקט Response הוא אובייקט מועיל נוסף ב-ASP. קודם התייחסנו לפקודה Response.Write המשמשת להעברת HTML וטקסט מסוגים אחרים, חזרה לדפדפן.

הפקודה Response.Write

ניתן לבנות מחרוזת שתועבר בפקודת Response.Write תוך שימוש באופרטור השרשור &, כמתואר בדוגמה הבאה :

```
<%
	Response.Write ("Hello, " & Session("USERID") & "<BR>")
%>
```

הערה:

ניתן גם להשתמש בתגי קיצור של Scripts ובתו השווה (=) כדי להחזיר ערך משתנה לדפדפן, כפי שהדבר נעשה בדוגמה הבאה:

```
Hello, <% =Session("USERID") %> <BR>
```

אני מעדיף את השימוש בפקודה Response.Write, אך עדיף להשתמש בתחביר המקוצר, אם יש לצרף ערך יחיד לקבצי HTML.

בנוסף לכתיבת קוד HTML, הפקודה Response.Write יכולה לשמש גם ליצירת Scripts שיפעלו ביישום הלקוח, תוך עבודה מהשרת. לדוגמה, נתייחס לקטע הבא:

```
<%
	sString = "This is an ASP variable"
	Response.Write (vbCrLf & "<SCRIPT Language=VBScript>" &vbCrLf)
	Response.Write ("MsgBox " & Chr(34) & sString & Chr(34) & vbCrLf)
	Response.Write ("</SCRIPT>" & vbCrLf )
%>
```

קטע הקוד מחולל פקודות VBScript שתבוצענה על ידי הדפדפן. פקודות Response.Write תתבצענה בשרת ותחזרנה לדפדפן את תגי <SCRIPT> ואת התוכן התחום ביניהן. כלומר ניתן להשתמש בקוד בדפי ASP כדי לחולל קוד לדפדפן. אם המשפט נראה לא מובן, הקלד קוד מקור זה, וצפה בתוצאה באמצעות דפדפן. כדאי שתשים לב לאופן הטיפול בתווי גרש, בקוד הלקוח, המבוצע כאילו הזנת אותו ידנית.

אחד השימושים ב-Script לקוח המחולל על ידי השרת הוא לשליטה על אובייקטי ActiveX המוטבעים בדפי Web. לדוגמה, ייתכן שדף ASP יכיל תג <OBJECT> של תיבת רשימה או פקד אחר, ובנוסף, פקודות המוסיפות להם פריטי מידע ספציפי.

אובייקט Response ושיטות אחרות

בדוגמה שעסקה בנושא אבטחה, למדת על שתי שיטות נוספות של אובייקט Response: Redirect ו-End. השיטה End משמשת לסיים את העברת התגובה הנוכחית ללקוח, כמו בדוגמה הבאה:

```
<H1> Here is Some HTML the browser will see! </H1>
<% Response.End %>
<H1> but the browser will never see this! </H1>
```

השיטה Redirect משמשת כדי לבקש מהדפדפן לעבור לכתובת URL אחרת:

```
<% Response.Redirect "http://www.callssomeonewhocares.com" %>
```

שורת הקוד פועלת רק אם לא הוחזר לדפדפן קוד HTML כלשהו, כלומר ששימוש בפקודת Response.Redirect לאחר פקודת Response.Write יגרום לשגיאה.

Expires הוא אחד המאפיינים של אובייקט Response. בוודאי ידוע לך שדפדפן מאחסן באופן זמני בדיסק כל דבר שהוא קורא מה-Web. אוסף הקבצים הזמניים הזה מכונה מטמון (Cache). יש סוגי קבצים כדוגמת קבצי תמונה, שמתאימים לאחסון במטמון ואילו סוגי קבצים אחרים (כגון קבצים הקשורים בניהול כניסת משתמשים למערכת) שאינם מתאימים לאחסון במטמון. במאפיין Expires יש להציב את מספר הדקות שאתה מעוניין שקבצים יהיו מאוחסנים במטמון. לדפים הקשורים באבטחת המערכת סביר להניח שתרצה להציב ערך 0 במאפיין הזה כבדוגמה הבאה:

```
Response.Expires = 0
```

אובייקט Response משמש גם להעברת עוגיות (Cookies) לדפדפן המשתמש. העברת העוגיות דומה להעברת משתני הפעלה, אך הן מאוחסנות במחשב המשתמש (IIS אף משתמש ב-Cookies לניהול המעקב אחר מזהי הפעלה). Cookies הם אמצעי נוח לשמירת הגדרות אישיות בדיסק המשתמש. אך לא תמיד ניתן להיות בטוחים שהן תמצאנה שם, משום שמשתמשים רואים בהן הפרת פרטיות ומנטרלים אותן. משלוח Cookie לדפדפן משתמש מבוצע על ידי הצבת ערך במשתנה הפעלה, באופן הבא:

```
Response.Cookies("MYCOOKIE") = 1234
```

באוסף Cookies דומה משתמשים עם אובייקט Request לאחזור ערכי Cookies.

אחזור נתונים באמצעות אובייקט Request

אובייקט Request מאפשר לקוד ASP לגשת לכל הנתונים הקשורים בבקשות המתקבלות מהדפדפן. כפי שידוע לך ניתן להיעזר באובייקט Request כדי לאחזר ערך פרמטר בכתובת URL על ידי שימוש בשם הפרמטר בשילוב עם מאפיין QueryString של האובייקט. בדומה לאוספים אחרים, גם ב-QueryString יש מאפייני Count ו-Index המאפשרים לעבור על כל הפריטים באוסף ולאחזר את ערכיהם:

```
Dim n
For n = 1 to Request.QueryString.Count
  Response.Write ("Parameter " & n & "Value = " &
            Request.QueryString(n) & "<BR>")
Next
```

לקבלת נתונים מטפסי HTML תוכל להיעזר גם באוסף Form של אובייקט Request. לדוגמה, אם יש בטופס בשדה בשם txtLastName, תוכל להציג את ערך השדה שהועבר לשרת על ידי שימוש בשורת הקוד הבאה:

```
Response.Write("You entered " & Request.Form("txtLastName")
```

למרבית הנתונים באובייקט Request ניתן לגשת על ידי שימוש באוספים. ניתן להשתמש בלולאת For...Each כדי להציג את רשימת כל האובייקטים באוסף. תוכנית נספח 4.7 תציג את כל המידע שבאוסף Request.ServerVariables.

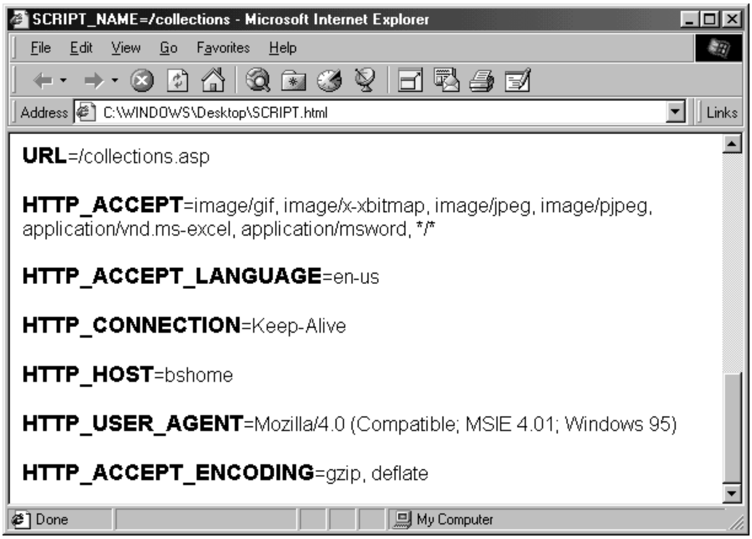
תוכנית נספח 4.7: VARIABLES.ASP - אוסף ServerVariables

```
<%@ LANGUAGE="VBSCRIPT" %>
<HTML>
<HEAD><TITLE>Collections</TITLE></HEAD>
<BODY>
<HR>
<H1 align="center">Server Variables</H1>
<%
For Each item in Request.ServerVariables
%>
<STRONG><%=item%></STRONG>=<%=Request.ServerVariables(item)%><BR>
<%
Next
%>
</BODY>
</HTML>
```

צור קובץ ASP חדש בשרת Web ושמור אותו בשם VARIABLES.ASP. העתק אליו את הקוד בתוכנית נספח 4.6. Script זה יציג את רשימת כל משתני השרת, כמו זו הנראית בתרשים נספח 4.6.

הערה:

ניתן להוסיף לולאות For...Each נוספות לקוד בתוכנית נספח 4.2 להצגת תוכן אוספים אחרים כגון Request.Form. אפשרות זו יכולה לסייע בניפוי שגיאות.



תרשים נספח 4.6: משתנה HTTP_USER_AGENT מאפשר לזהות את סוג דפדפן המשתמש

אובייקט Server

השימוש העיקרי באובייקט Server הוא ליצירת אובייקטים שיפעלו בשרת Web, כך שיישומי ASP יוכלו לגשת אליהם. כבר למדת ליצור אובייקט ADO Connection:

```
Set cn = Server.CreateObject("ADODB.Connection")
```

ניתן ליצור אובייקט ממחלקה שתהדר לקובץ ActiveX DLL, פעולה שתתואר בהמשך.

כתחליף לשימוש בפקודה Server.CreateObject תוכל להטביע בקובץ ASP תג <OBJECT>, כשהאפשרות RUNAT מוגדרת ל-Server:

```
<OBJECT RUNAT="Server" ID=cn PROGID="ADODB.Connection"></OBJECT>
```

שני התחבירים ייצרו אובייקט מסוג ADODB.Connection בשרת Web. אך חשוב לזכור שלא ניתן להשתמש בתגי <OBJECT> בתוך תגי Script.

אובייקט Application ו-GLOBAL.ASA

בתחילת הפרק למדת להיעזר במבנה עץ התיקיות לזיהוי יישומי ASP. יכולת זו חשובה לקישור אירועי Start ו-End בקוד התוכנית, עם יישומים והפעלות. כדי לכתוב קוד לאירועים יש לאחסן קובץ מיוחד בשם GLOBAL.ASA בספריית הבסיס (שורש) של יישום ASP. הקובץ יכיל קוד לטיפול באירועים אלה, כמתואר בדוגמה הבאה:

```
Sub Session_OnStart
   Dim x
   Set x = Server.CreateObject("ADODB.Connection")
   Set Session("Connection")
End Sub
```

תוכל להשתמש בקובץ GLOBAL.ASA לניהול מונה כניסות לאתר, פתיחה וסגירת קבצים, או כל פעולה אחרת האמורה להתבצע בעת פתיחה או סגירת יישום.

ActiveX DLL ב-ASP

קוד VBScript בדפי ASP מאפשר לבצע פעולות רבות, אך VBScript חסרה כמה מיכולות Visual Basic, וכן את יכולות סביבת הפיתוח המשולבת - IDE (Integrated Development Environment) שבהן אתה רגיל להשתמש. אך על ידי שימוש בפונקציה Server.CreateObject תוכל להגדיר מופעי ActiveX DLL בדפי ASP.

תחילה יש ליצור את ActiveX DLL בסביבת הפיתוח הרגילה של Visual Basic ולנסותו. אחר יש להתקין ולרשום את קובץ DLL בשרת Web, כדי לאפשר שימוש בו בדפי ASP. יצירת קבצי DLL תוארה בפרק 16, **מחלקות: שימוש חוזר ברכיבים**.

ראה, "יצירת קבצי ActiveX DLL", פרק 16.

לאחר שתיצור את קובץ ActiveX DLL תוכל להתקין אותו בשרת Web. אם כבר התקנת את קבצי Runtime של Visual Basic בשרת, כל שיש לעשות כדי להתקין את קובץ DLL, הוא להעתיק אותו לשרת ולהפעיל את התוכנית REGSVR32, כדי לרשום אותו ברישום.

לאחר שתתקין ותרשום את קובץ DLL, תוכל ליצור אובייקט שישתייך למחלקה שהוגדרה בקובץ DLL, על ידי שימוש בפקודה Server.CreateObject:

```
Set objVariable = Server.CreateObject("MyProject.MyClass")
```

לאחר שתיצור את האובייקט, תוכל לגשת אליו מדף ASP הנוכחי. אם תרצה להעביר את האובייקט לדף אחר, תוכל לעשות זאת על ידי אחסונו אותו במשתנה הפעלה, כפי שנעשה בדוגמת שימוש בקובץ GLOBAL.ASA שהובאה בסעיף הקודם.

השימוש בקבצי ActiveX DLL בדפי ASP הוא פשוט מאוד, כל שיהיה עליך לעשות הוא לבצע פניות לשיטות ולמאפיינים באמצעות קוד VBScript:

```
Response.Write objVariable.GetUserData("Smith")
```

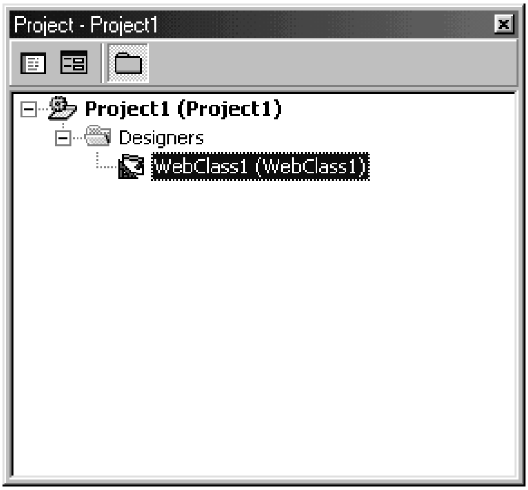
זכור שכל המשתנים המשמשים בקוד VBScript הם מסוג Variant, כך שיש לנהוג בזהירות בעת ההצהרה על הפונקציות שתיכללנה בקבצי DLL. האופנים שבהם יפעלו מערכים וסוגי אובייקטים נוספים ב-VBScript עשויים להיות שונים מאופני פעולתם ב-Visual Basic. אני ממליץ להגדיר פונקציות ניסוי קטנות בקובץ DLL, שתצגנה את מגבלות VBScript בתחום זה.

פרויקט יישום IIS

IIS Application (יישום IIS) הוא סוג פרויקט חדש ב- Visual Basic. IIS Application הוא סוג מיוחד של ActiveX DLL, הפועל בשרת Web ומעבד בקשות שמתקבלות מיישומי לקוח ומחזיר קוד HTML לדפדפן המשתמש. אם אופן הפעולה נשמע לך דומה לאופן הפעולה של דפי ASP, צדקת. יישומי IIS Applications מאפשרים לבצע גישה לאובייקטים רבים אשר ASP מאפשר לגשת אליהם (Request ,Response ,Server ועוד), ויישומים אלה גם "מתארחים" לעיתים בדפי ASP. למעשה אופן הפעולה דומה מאוד לאופן הקריאה לקבצי ActiveX DLL שמיושם מתוך דפי ASP. ההבדל הוא שיישום IIS נועד מראש לעבודה ב-Web. ActiveX DLL מכיל מודול מחלקה רגיל של Visual Basic, ו-IIS Application מכיל **WebClass** (מחלקת Web).

יצירת יישום IIS

כדי להתחיל ליצור יישום IIS הפעל את Visual Basic ובחר IIS Application מרשימת סוגי הפרויקטים. כך תיצור פרויקט חדש שיכיל אובייקט יחיד בשם WebClass1 כמתואר בתרשים נספח 4.7.



תרשים נספח 4.7: יישום IIS הוא סוג קובץ ActiveX DLL מיוחד שפועל בשרת Web

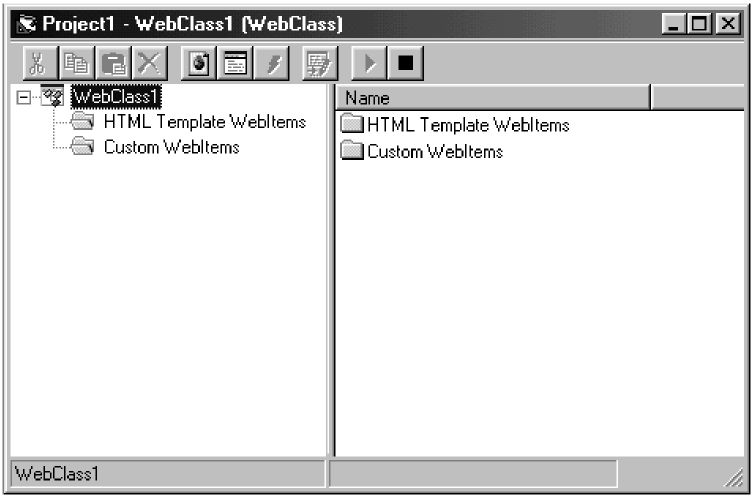
WebClass1 אינו טופס או מודול מחלקה סטנדרטי, הוא **ActiveX Designer** (כלי לעיצוב ActiveX). ActiveX Designers מסייעים ביצירת סוגים מיוחדים של רכיבי ActiveX והם מהווים חלק אינטגרלי של סביבת הפיתוח ב- Visual Basic. בעת עבודה עם יישומים מסוג IIS Application, WebClass Designer מבצע פעולות "בסיסיות" רבות הקשורות בעבודה עם HTML. למעשה כאשר תהדר את קובץ DLL שיכיל את היישום, רוב הקוד ישמש ל"דיבור" עם WebClass DLL.

משום שהמבנה בסיסי של ה-Web הוא מבנה מבוסס מסמכים (Document Based), פעולתו התקינה של יישום IIS תלויה בכך שכל קובץ יימצא במקום בו הוא אמור להיות. על כן יש להקפיד לשמור את הפרויקט, בטרם תבצע עליו פעולות כלשהן. כדאי שתגדיר ליישום תיקיה נפרדת ושתשמור בה את קבצי הפרויקט.

כמו עם טפסים, גם בעבודה עם WebClass מוצעות תצוגת קוד (Code View) ותצוגת עיצוב (Design View). פתח את תצוגת העיצוב על ידי לחיצה ימנית על WebClass ובחירה בפקודה **View Object**. יוצג המסך הנראה בתרשים נספח 4.8.

בתרשים נספח 4.8 ניתן לראות ש-WebClass מכיל WebItems (פריטי Web). קיימים שני סוגי WebItems:

- HTML Template WebItem (תבניות קבצי HTML) - WebItem המבוסס על תבנית קובץ HTML קיימת.
- Custom WebItem - WebItem שמוגדר כולו בקוד.

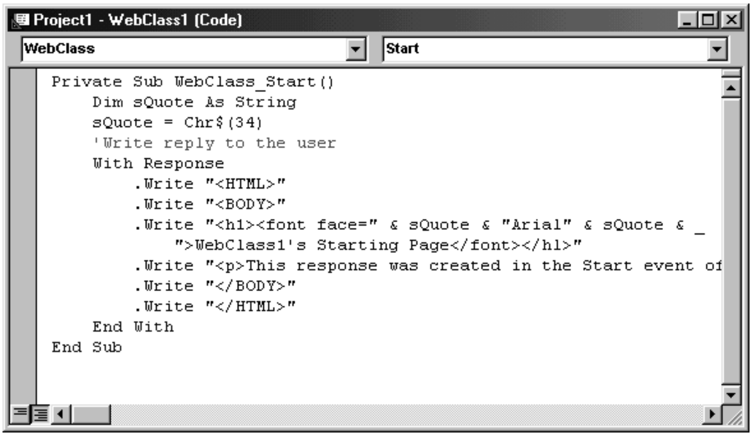


תרשים נספח 4.8: WebClass יכול להכיל שני סוגי Web Items (פריטי Web)

מהם WebItems? WebItems הם קטעי קוד (זכור שהם בקבצים מסוג ActiveX DLL והיות שהם יישומים המיועדים לפעול בשרת הם חסרי ממשק חזותי כלשהו. WebItems מהווים אמצעי קישור בין אירועי תוכניות Visual Basic לבין העולם מבוסס המסמכים של ה-Web.

הפעלת יישום IIS

לפני שנמשיך, כדאי שנעיין בקוד WebClass. לחץ לחיצה ימנית על שם המחלקה בחלון Project Explorer ובחר את הפקודה View Code מהתפריט שיוצג. יוצג קוד ברירת המחדל של אירוע Start של המחלקה WebClass, שמוצג בתרשים נספח 4.9.



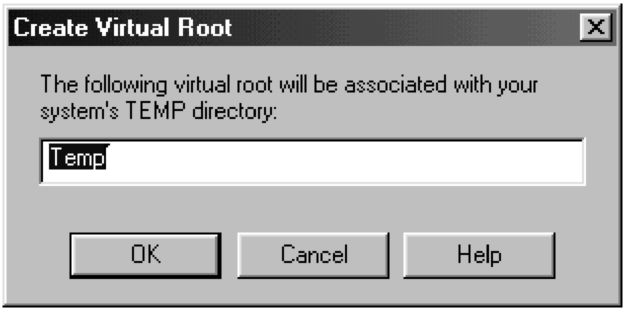
תרשים נספח 4.9: אירוע Start של המחלקה WebClass מגדיר מסך HTML התחלתי תוך שימוש באובייקט Response

הפעל את הפרויקט באמצעות לחיצה על לחצן Start או על ידי הקשה על F5. זכור שמשום שבדומה ליישומי ASP, גם יישומי IIS הם יישומים מבוססי Web, יידרש שרת Web פעיל כדי שתוכל להריץ אותם או לנפות מהם שגיאות.

הערה:

הדוגמאות שתוצגנה בפרק נוצרו ב- Visual Basic **תוך שימוש ב-** Personal Web Server **וב-** Transaction Server **(שנכלל ב-** Personal Web Server**) שפעלו על אותו מחשב.**

בכל פעם שתפעיל את יישום IIS בסביבת הפיתוח המשולבת של Visual Basic, התוכנה תיצור קובץ ASP ותנסה לפתוח אותו באמצעות Internet Explorer. בפעם הראשונה שתפעיל יישום IIS חדש, תוצג הודעה כמו זו המוצגת בתרשים נספח 4.10, שתבקש להזין שם תיקיה מדומה.



תרשים נספח 4.10: Visual Basic **יוצרת באופן אוטומטי תיקיה מדומה וקובץ** ASP **שישמשו לאירוח** WebClass

כאשר היישום ירוץ, יוצג מסך HTML שנוצר באירוע Start של מחלקת WebClass. אם תעיין בתיקיית הפרויקט, ייתכן שתבחין כי Visual Basic יצרה קובץ ASP שייצור מופע של מחלקת WebClass. סביר להניח שכעת אופן הפעולה של יישום IIS מובן לך מעט יותר - הוא דומה לאופן הפעולה של קובץ ASP, אך כל הגישה ל-Web מתבצעת על ידי שימוש בסביבת Visual Basic. בסעיף הבא תלמד מעט יותר על יכולות מחלקת WebClass, באמצעות הוספת WebItems למחלקת WebClass. לעת עתה יהיה עליך לעצור את פעולת הפרויקט ולשוב לסביבת הפיתוח של Visual Basic.

מופעי WebClass

פתח שוב את חלון Code ועיין ברשימת האירועים הנתמכים על ידי מחלקת WebClass. הוסף לכל שגרת אירוע שורת קוד בנוסח Debug.Print Eventname והרץ שוב את הפרויקט. שים לב לרצף האירועים הבא:

1. כאשר תתחיל ליצור את הפרויקט, יופעל הדפדפן. קובץ ASP שייטען לדפדפן יכיל פקודת Server.CreateObject, שתיצור מופע של המחלקה WebClass ותפעיל את אירוע Initialize.
2. קובץ ASP יעביר למחלקת WebClass בקשה לקבלת דף Web, שתגרום להפעלת אירוע BeginRequest.
3. אירוע WebClass Start יופעל (בדומה לאופן בו תופעל שגרת האירוע Form_Load) ויגרום למשלוח קוד HTML חזרה לדפדפן.
4. השלמת משלוח קוד HTML לדפדפן תהווה את השלמת מילוי הבקשה, מצב שיגרום להפעלת אירוע EndRequest.
5. אירוע Terminate יופעל, ויגרום להריסת מופע המחלקה WebClass.

ייתכן שפעולה מספר 5 גרמה לך לתמוה מדוע מופע המחלקה WebClass נהרס מייד לאחר יצירתו? התשובה היא משום שמאפיין StateManagment של אובייקט WebClass הוגדר wcNoState. הגדרה כזו גורמת לכך שאם הגדרת משתנה מקומי או מאפיין באירוע WebClass Start, ההגדרה לא תהיה נגישה במהלך הקריאות הבאות לאובייקט. אם מאפיין WebClass מוגדר wcRetainInstance, כל הקריאות לאובייקט WebClass שמקורן בדף ASP מתבצעות לאותו מופע האובייקט, ותוכל לאחסן נתוני מצב.

אך בפעולה זו טמונה התייחסות לנושא בסיסי יותר מסתם הגדרת מאפיין. הנושא הוא יותר בחזקת פילוסופיה של כיצד לכתוב יישומים. האופי הבלתי מקושר של ה-Web מתאים יותר לביצוע עיבוד מבוסס טרנזקציות מאשר לעיבוד מבוסס מצב. בעבודה עם טרנזקציות, אינך צריך להתייחס לתקשורת עם מופע האובייקט, מפני שכל המידע הדרוש להשלמת הטרנזקציה מועבר בכל קריאה לאובייקט.

מרביתנו נוהגים לכתוב קוד שתלוי בכך שהאובייקט במצב מסוים. למשל, תוכל לכתוב קוד שיגדיר מספר מאפייני אובייקט ואחר יבצע קריאה לשיטה שתפעל על האובייקט תוך שימוש במאפיינים שהגדרת.

```
x.AccountID = 123456
x.AccountHolder = "Stephanie and Brent"
x.Amount = "$20.00"
x.DepositMoney
```

בקטע קוד זה, x מייצג מופע של רכיב ActiveX המטפל בפונקציות הקשורות בניהול חשבונות בנק. השורה האחרונה בקוד היא קריאה לשיטה DepositMoney (הפקד כסף), שסביר להניח שנדרשים לה שאר נתוני החשבון כדי שהיא תוכל לפעול. קטע קוד זה

מתבסס על אפשרות לגישה חוזרת לאובייקט. ניתן לכתוב את קטע הקוד גם בשורה אחת שבה מועברים לאובייקט כל הנתונים הדרושים:

```
x.DepositMoney(123456, "Stephanie and Brent", "$20.00")
```

הדוגמה שהובאה פשוטה מאוד, אך נסה לחשוב רגע על דוגמה מורכבת שבמהלכה מתבצעות קריאות חוזרות ונשנות לפונקציות האובייקט. בדוגמה הראשונה יש גישה למופע אובייקט x, אך בדוגמה השנייה אין משמעות לנושא הגישה למופע מסוים. הדוגמה השנייה אומנם עושה רושם שלמימושה יש להשקיע יותר מאמץ, אך היא ניתנת למימוש ללא תלות במצב המערכת. ניתן להשתמש בכלים כגון Transaction Microsoft Server לאגור את קישורי האובייקט ולייצור יישומים מדורגים.

WebItem של תבניות HTML

כדי ללמוד על אופן הפעולה של HTML Templates (תבניות קבצי HTML), ניצור תחילה תבנית פשוטה של קובץ HTML בשם TEST.HTM:

```
<HTML>
<BODY>
<H1>This is a test page</H1>
<BR>
<A HREF=http://www.somewhere.com> Link to somewhere!</A>
<IMG SRC=guestbook.gif>
</BODY>
</HTML>
```

הקובץ TEST.HTM נראה כקובץ חסר ייחוד עד שתייבא אותו לפרויקט. כדי לייבא, פתח את חלון העיצוב של WebClass1, לחץ לחיצה ימנית על המחלקה WebClass1 ובחר מהתפריט שיוצג את הפקודה **Add HTML Template**. בחר את הפריט TEST.HTM מתיבת הדו-שיח File. חלון עיצוב ייראה כמו בתרשים נספח 4.11.

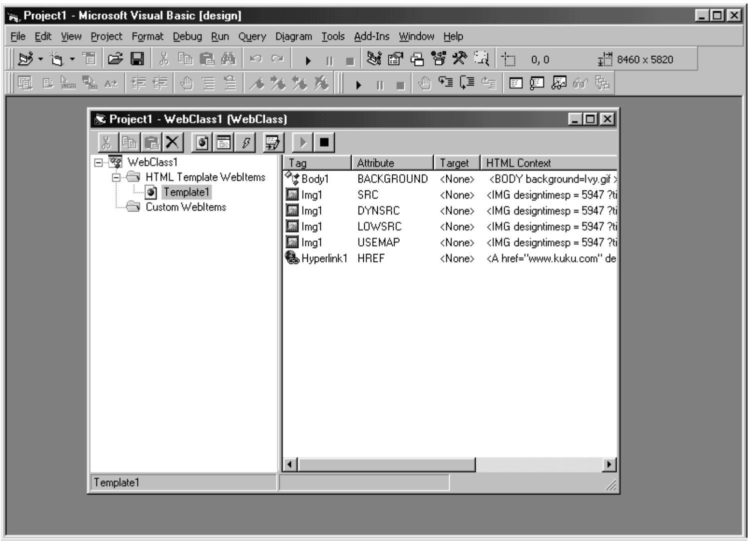
הערה:

כדאי לייבא את קובץ התבנית לפרויקט, מתיקיה שאינה תיקיית הפרויקט כי Visual Basic שומרת באופן אוטומטי עותק של קובץ התבנית בתיקיית הפרויקט.

לאחר ייבוא קובץ HTML הוא הופך ל-WebItem באופן אוטומטי.

הגדרת אירוע קובץ תבנית

בחלונית הימנית של חלון העיצוב תוצג רשימת כל תגי HTML שבקובץ התבנית, שניתן לקשר לאירועי Visual Basic. WebItem החדש שיצרת קיבל את השם Template1.



תרשים נספח 4.11: לאחר הוספת WebItem ל-WebClass תוצג רשימת כל התגים בתבנית שניתן לקשר לאירועי Visual Basic

לשם השוואה נתייחס למה שמתרחש כאשר מוסיפים פקד TextBox לטופס רגיל של Visual Basic. במצב כזה אתה ממקם אובייקט (הפקד) באובייקט מכיל (Container Object) - הטופס. לאחר שתשרטט את הפקד, תוכל לגשת לכל המאפיינים והאירועים שלו באמצעות חלון Code של הטופס. הקשר הקיים בין המחלקה WebClass לאובייקט Template1 דומה לקשר הקיים בין הטופס לפקד תיבת הטקסט.

ההבדל היחיד שקיים בין אירועי תיבת טקסט לאירועי WebItem, הוא שאירועי WebItem אינם זמינים מתוך חלון הקוד באופן אוטומטי. תחילה יש לקשר את התגים המתאימים לאירועי התוכנית.

נסה לעשות זאת בתוכנית הדוגמה. היזכר בקישור לאתר www.somewhere.com שהגדרת בקובץ התבנית, כאשר ייבאת את קובץ התבנית לפרויקט, הקישור קיבל את השם Hyperlink1. לחץ לחיצה ימנית על הפריט Hyperlink1 ובחר את הפקודה Connect to Custom Event. כך יוצג בחלון העיצוב אירוע חדש, שגם הוא יקבל את השם Hyperlink1. לחץ לחיצה כפולה על שם האירוע וחלון הקוד ייפתח בקטע קוד האירוע Template1_Hyperlink1. אירוע זה יופעל בכל פעם שהמשתמש ילחץ על הקישור. הוסף לאירוע את פקודת Response.Write המוצגת בקטע הקוד הבא:

```
Private Sub Template1_Hyperlink( )
  Response.Write "<BR> You Clicked me!<BR>"
End Sub
```

הפעל כעת את הפרויקט. לאחר שהדף הראשון ייטען, שנה את שם הקובץ המוצג בסוף כתובת URL (השם ההתחלתי יהיה Project1_WebClass1.ASP או שם דומה) לשם קובץ התבנית TEST.HTM והקש על Enter. דף התבנית ייטען. אם תבחר את הפקודה View Source, תראה שהקישור כבר אינו מצביע על האתר www.somewhere.com ולא על אירוע במחלקה WebClass. לחץ על הקישור ואם הכל תקין, המילים You Clicked Me תוצגנה בחלון הדפדפן. בכך הדגמנו כיצד ניתן לקשר Hyperlink בקובץ HTML לאירוע תוכנית Visual Basic.

משלוח תבנית לדפדפן

כברירת מחדל, אירוע Start של המחלקה WebClass שולח מספר שורות HTML לדפדפן. ניתן גם לשלוח תבניות, כדוגמת זו שיצרת זה עתה, על ידי שימוש בשיטה WriteTemplate. פתח את שגרת אירוע Start של המחלקה WebClass והחלף את שורות הקוד שנכללו בשיגרה כברירת מחדל בשורה באה:

```
Me.Template.WriteTemplate
```

הפעל את היישום, והדבר הראשון שיוצג ביישום יהיה קובץ תבנית המבוסס על הקובץ TEST.HTM. שים לב שהדפדפן אינו מנותב לקובץ TEST.HTM ובמקומו, קוד HTML שבקובץ התבנית יועבר באירוע Start.

השיטה WriteTemplate עשויה להיות כלי מועיל מאוד, כאשר משלבים אותה עם אירוע ProcessTag. האירוע יכול לשמש ליצירת קוד HTML באופן דינמי, דומה לזה שמיושם על ידי פקודות Response.Write בקבצי ASP. השיטה פועלת על ידי חיפוש אחר תגים בקובץ התבנית (שדומים מאוד לתגי HTML אך כוללות גם תחילית מיוחדת) ומחליפה אותם בנתונים שתספק.

כדי להמחיש זאת, הצג כעת את קובץ התבנית TEST.HTM (ודא שאתה מציג את העותק שבתיקיית הפרויקט ולא את קובץ התבנית המקורי), הוסף את שורת הקוד שלהלן במקום כלשהו בקטע Body של הקובץ ושמור את הקובץ המעודכן:

```
<WC@mytag>My Information</WC@mytag>
```

שורת הקוד מכילה תג "מותאם" <WC@mytag>. במהלך פעולת השיטה WriteHTMLTemplate, בכל פעם שהמחלקה WebClass נתקלת בתג עם התחילית @WC, היא מפעילה אירוע ProcessTag. שם התווית (mytag) ותוכנה (My Information) מועברים לאירוע וניתן לשנותם לפני העברתם לדפדפן.

```
Private Sub Template1_ProcessTag(ByVal TagName As String, TagContents As String,
SendTags As Boolean)
If TagName = "WC@mytag" Then
    TagContents = "<B> Here is some new information </B>"
End If
End Sub
```

WebItem מותאם אישית

Custom WebItem (WebItem מותאם אישית) הוא הסוג השני של WebItem שאותו ניתן לצרף ל-WebClass. אירועי Custom WebItems הם כמו אירועי HTML Template WebItems וניתן להוסיף להם גם אירועים משלך, שאליהם ניתן לפנות מתוך דפי Web. אך ב- Custom WebItem לא ניתן לשלב תבניות. WebItems כאלה קיימים רק בקוד שלך, ומתקיימים בצורת אוסף שגרות אירוע. ניתן לקשר פריטים ב- WebItem מסוג HTML Template ל- Custom WebItem על ידי ביצוע מספר פעולות פשוטות:

1. לחץ לחיצה ימנית על WebClass בחלון Project Explorer ובחר את הפקודה View Object, כדי לפתוח את Designer Window (חלון עיצוב).
2. לחץ שוב לחיצה ימנית על שם המחלקה ובחר את הפקודה Add Custom WebItem.
3. בחר שם תג HTML ב- HTML Template WebItem וקשר אותו ל- Custom WebItem.
4. כתוב קוד עבור האירוע באופן שתואר בסעיף הקודם.

כפי שבוודאי הבחנת במהלך ההתייחסות לדוגמאות, השימוש ב-WebClasses הוא מעט מורכב יותר מהשימוש בדפי ASP רגילים, אך הם פעילים בסביבת Visual Basic ופותחים מיגוון רחב של אפשרויות חדשות.

מכאן...

בפרק זה למדת על ASP, שהם אמצעי המאפשר ליישם את הידע ב- Visual Basic באינטרנט. מידע על נושאים אחרים הקשורים ב-Web תוכל למצוא בפרקים הבאים:

- בפרק 14 **יצירת פקדי ActiveX** תוכל ללמוד על יצירת פקדי ActiveX הניתנים לשימוש בדפי Web.
- בפרק 31 **מסמכי ActiveX** תלמד על שיטה חדשה להסבת תוכניות Visual Basic לפעולה בסביבת Web.
- מידע על נושאים נוספים הקשורים ב-Web תוכל למצוא בפרק 32 **Visual Basic ושימושים נוספים באינטרנט**.

נספח 5

התקליטור המצורף

מה בתקליטור:

- **קטלוג HTML** קטלוג ספרי המחשבים האינטראקטיביי של הוצאת הוד-עמי.
(לא נדרשת התקנת תוכנה. מומלץ לצפייה עם Internet Explorer מגירסה 4 ומעלה).
- **מילון הוד-עמי למונחי מחשב** בשיתוף **מכון התקנים הישראלי** המכיל כ- 5,800 ערכים מתחום טכנולוגיית המידע.
- **גרסת הלימוד של שפת הפיתוח Visual Basic 6 - Visual Basic 6 Working Model**
- **מספר תוכנות עזר שימושיות.**
- **קבצי תרגול.**

הערה:

אם מנהל התקן כונן התקליטורים המותקן הוא 16 סיביות - ייתכן ותראה רק 8 תווים ראשונים של שם הקובץ (במקרה ובמקור הוא ארוך יותר).

הסיבה: כונני תקליטורים במהירות x4 עובדים עם מנהל התקן שעבד בסביבת DOS ו-Windows 3.11 ויכול לעבוד גם עם Windows 95, למעט היכולת לזהות קבצים עם שמות ארוכים.

הפתרון: להתקין מנהל התקן 32 סיביות (אם קיים) או לקנות כונן תקליטורים חדש ולוודא שמצורף אליו מנהל התקן 32 סיביות.

התקנה - Acrobat Reader

יש להתקין תוכנה זו כדי לקרוא ולהדפיס את הפרקים לדוגמה, אליהם ניתן לגשת באמצעות **קטלוג HTML** (שהתקנתו תוסבר בהמשך). התוכנה גם מאפשרת חיפוש בעברית ובאנגלית במסמך המוצג. בנוסף, בעזרת תוכנה זו תוכל לקרוא את המסמכים שההוצאה מפרסמת באתר האינטרנט. התוכנה פועלת במערכות הפעלה **Windows 95/98 בלבד!**

1. לחץ על לחצן **התחל** ובחר באפשרות **הפעלה**.
2. בתיבת הטקסט הקלד את הפקודה **X:\Software\Adobe Acrobat\Arme4ENU.exe** (החלף את האות X באות המייצגת את כונן התקליטורים שלך) ולחץ על **אישור**.
3. אשף ההתקנה מתקין את הרכיבים הנדרשים. עליך ללחוץ על **Next**, **Accept** ו-**Next** פעם נוספת כדי לסיים את ההתקנה.
4. בסיום ההתקנה עשויה להופיע על המסך תיבת דו-שיח **התנגשות בין גירסאות** ומייד אחר כך להיעלם. במקומה תופיע על המסך תיבת הודעה של תוכנית ההתקנה. לחץ על **אישור** ובתיבת הדו-שיח **התנגשות בין גירסאות** ששבה להופיע לחץ על **כן**, כדי לשמור את גרסת הקובץ שלך.

קטלוג HTML –

הוצאת הוד-עמי גאה לבשר על **קטלוג HTML** העושה שימוש בטכנולוגיות אינטרנט מתקדמות כדי להביא לך את המידע על ספרי המחשבים המקצועיים שלנו בלחיצת עכבר.

מומלץ לצפייה בעזרת Microsoft Internet Explorer מגירסה 4 ומעלה. -

בעזרת **קטלוג HTML** תוכל:

- לעיין במידע על ספרי ההוצאה מתי שתרצה (לחיצה כפולה.....וזהו!). -
- לעבור במהירות ובקלות בין הקטלוג והיישום בו אתה עובד. -
- לעיין במידע על כל ספר וספר. -
- לצפות ואף להדפיס פרק לדוגמה. -
- לגשת במהירות, בגישה אינטואיטיבית, תוך התמקדות מהירה בספר המבוקש. -
- לעיין בקטלוג בקצב אישי שלך. -
- לנווט את דרכך בקטלוג ולחזור ולהתרענן בכל נושא בכל רגע. -

הקטלוג מומלץ לצפייה בעזרת Internet Explorer מגירסה 4 ומעלה.

1. הכנס את התקליטור לכונן.
2. לחץ **התחל** ובחר **הפעלה**.
3. בעזרת לחצן עיון סמן את הקובץ **Setup.exe** אשר בתיקיה הראשית של התקליטור המצורף.
4. לחץ **פתח**.
5. לחץ **אישור**.

המחירון המעודכן של ספרי ההוצאה נמצא באתר האינטרנט www.hod-ami.co.il


קטלוג ספרי מחשבים בהוצאת הוד-עמי

1. ודא שתקליטור הוד-עמי נמצא בכונן התקליטורים.
2. הפעל את הסמל עם הכיתוב **קטלוג ספרי מחשבים בהוצאת הוד-עמי** שעל שולחן העבודה.

מילון הוד-עמי למונחי מחשב

בשיתוף מכון התקנים הישראלי

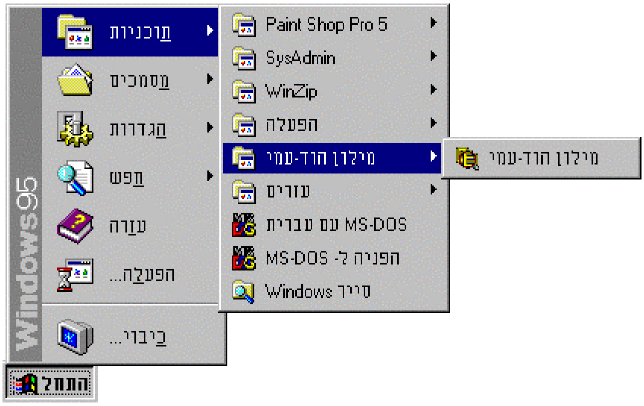
מילון הוד-עמי למונחי מחשב בשיתוף **מכון התקנים הישראלי** נועד לתת בידי אנשי המקצוע בתחום המחשוב ובידי משתמשים אחרים את אוסף המונחים והביטויים המקובלים ביותר בתחום זה באנגלית **ותרגומם לעברית**.

התקנה

1. הכנס את התקליטור לכונן.
2. לחץ על **התחל** ובחר **הפעלה**.
3. בעזרת לחצן **עיון** סמן את הקובץ **HodAmiDictionary.bat** אשר בתיקיה הראשית של התקליטור המצורף.
4. לחץ **פתח**.
5. לחץ **אישור**.
6. הקש **Enter** לכל שאלה שתישאל במהלך ההתקנה (אל תשנה את ספריית ברירת המחדל המוצעת).

הפעלה

לחץ על **התחל** ובחר **תוכניות**, **מילון הוד-עמי**, **מילון הוד-עמי**.



הטיפול המקיף במונחי השפה העברית מופקד בידי **האקדמיה ללשון העברית** ובידי **אגף התקינה במכון התקנים הישראלי**. כדי לענות לצרכי עולם המחשוב הדינמי חברו **הוצאת הוד-עמי** ו**מכון התקנים הישראלי** כדי להפיק ולהוציא לאור את המילון שלפניכם.

התשתית למילון היא סדרת **התקנים הישראליים ת"י 1080 למונחי מחשב**, המבוססים על התקנים של ארגון התקינה הבינלאומי **ISO**. מונחים אלה מסומנים בלוגו של מכון התקנים, כפי שמוסבר במסך העזרה. המילון כולל גם מונחים שהוגדרו על ידי מוסדות וגופי הדרכה שונים, על ידי מיקרוסופט ישראל, מונחים שפורסמו בעתונות ובספרות מחשבים בעברית, וגם מונחים המקובלים על ידי משתמשים (גם אם לפעמים אינם מומלצים, אך מושרשים) ועוד. המילון עוסק במונחים המקובלים במידה שהם נמצאים בשימוש כללי. מונחים מעטים לא תורגמו ונשארו במקורם האנגלי.

המילון הממוחשב מכיל כ- **5,800** מונחים. הוא מאפשר חיפוש מונחים באנגלית ובעברית, או מחרוזות תווים (כמו צירופי מילים) בכל אחד משדות הנתונים: מונח עברי, מונח אנגלי, הגדרה, הערה או כל השדות. המילון הממוחשב מאפשר דפדוף והדפסה.

מילון הוד-עמי למונחי מחשב יצא לאור בדפוס ברבעון הראשון של שנת 2000 ויכיל את מילון המונחים, אינדקס מונחים בעברית ואינדקס ראשי-תיבות באנגלית.

כדי לשמור על **עדכניות** המילון, **הוצאת הוד-עמי** תשמח לשתף פעולה ולקבל הערות והצעות מכל מי שמתעניין בנושא. כל הערה, תיקון או תוספת יתקבלו בברכה.

הערות והצעות למילון הוד-עמי למונחי מחשב:

הוצאת הוד-עמי לספרי מחשבים
מילון מונחי מחשב
ת.ד. 6108, הרצליה 46160
טלפון: 09-9564716
פקס: 09-9571582
דואר אלקטרוני: sls@hod-ami.co.il

התקנת Visual Basic 6 Working Model

התקנת Visual Basic 6 Working Model דורשת כי דפדפן האינטרנט Internet Explorer יהיה מותקן במחשב (בגירסה 4 ומעלה). לצורך התקנת **Visual Basic 6 Working Model** עליך **להסיר** את התקנת Internet Explorer גירסה 3, אם קיימת, ורק אז לבצע את התקנת Visual Basic 6 Working Model.

לתשומת לבך: במידה ובמחשב היתה מותקנת גירסה קודמת של Internet Explorer צפויה במהלך ההתקנה להופיע תיבת דו-שיח המורה על **התנגשות בין גירסאות**. לשאלה **האם ברצונך לשמור את הקובץ הקיים במחשבך** השב **כן** וההתקנה תמשיך כרגיל.

התוכנה ניתנת כבונוס ללא תשלום לרוכשי הספר **הוצאת הוד-עמי אינה מספקת תמיכה במוצר.**

התקנת Visual Basic 6 Working Model מתבצעת באופן הבא:

1. לחץ על **התחל**, **הפעלה**, **עיון**.
2. עבור לכונן התקליטורים.
3. פתח את התיקיה **SoftWare**.
4. פתח את התיקיה **Visual Basic 6 Working Model**.
5. בחר בקובץ **Setup.exe** ולחץ על **פתח**.
6. בתיבת הדו-שיח **הפעלה** לחץ על **אישור** וההתקנה תחל.
7. במסך הראשון: **Visual Basic 6 Working Model** לחץ על **Next**.
8. במסך השני: **End User License Agreement** בחר: **I accept the agreement** ולחץ על **Next**.
9. במסך השלישי: **Product Number and user ID** הקלד את שמך ולחץ על **Next**.
10. במסך **Install DCOM98** ודא שתיבת הסימון מסומנת ולחץ על **Next**.
11. לאחר התקנת הרכיב **DCOM98** תוכנית ההתקנה תודיע שכעת יש לאתחל את המחשב מחדש. לחץ על **OK**.
12. לאחר שהמחשב יעלה מחדש, תימשך ההתקנה. המסך הבא שיופיע הוא **Choose Common Install Folder**. בדוק אם המיקום והנתיב שמציעה תוכנית ההתקנה מתאימים לך. אם לא, לחץ על **Browse** ובחר מיקום אחר. כעת כשהמיקום והנתיב מתאימים, לחץ על **Next**.
13. במסך: **Visual Basic 6 Working Model Setup** תופיע הודעה שיש לסגור כל יישום פתוח (למעט תוכנית ההתקנה), עשה זאת (באמצעות לחיצה על לחצן היישום שבשורת המשימות) ולחץ על **Continue**.

14. בשלב הבא לחץ על **OK** והמתן בעת שתוכנית ההתקנה מחפשת רכיבים מותקנים.

15. במסך הבא שיופיע בחר בסוג ההתקנה. אם יש לך מספיק מקום בדיסק שבחרת בחר בהתקנת **Typical**.

16. אם תוכנית ההתקנה תבקש להחליף Dll ענה **No** ואחר כשתשאל אם ברצונך להשאיר את הקבצים כמות שהם ענה **Yes**.

17. בשלב זה תתבקש לאתחל את Windows מחדש, לחץ על **Restart Windows**.

18. כשהמחשב "יעלה" מחדש תופיע תיבת דו-שיח המאפשרת רישום עותק התוכנה באמצעות האינטרנט. פעל לפי ראות עיניך.

ספר זה מבוסס על גרסת Enterprise של Visual Basic 6 ומתאים לעבודה גם בגרסאות אחרות, למעט גרסת Working Model (קוד המקור לא נבדק בגירסה זו).

היכן נמצאות התוכניות של ספר זה?

התיקיה הרלוונטית לספר זה: **59221** -

תחת תיקיה **X:\Books\59221** תמצא תיקיות משנה: תיקיה עבור כל פרק: תיקיה Chap02 עבור פרק 2, תיקיה Chap03 עבור פרק 3 וכך הלאה. בכל תיקיה נמצאים קטעי קוד המקור שמופיעים בספר.

בתיקיות אחרות תחת התיקיה Books נמצאים קבצים הרלוונטיים לספרים אחרים של ההוצאה.

העתקת קבצי המקור לדיסק

הקבצים נמצאים בתיקיה **X:\books\59221** מסודרים לפי תיקיות - תיקיה עבור כל פרק. כדי להעתיק את הקבצים לדיסק:

1. לחץ על לחצן **התחל**, **תוכניות**, **סייר Windows**.
2. הצג את תוכן התיקיה **Books** אשר בתקליטור.
3. סמן את התיקיה **59221** וגרור אותה לתיקיה כלשהי בדיסק.

ביטול המאפיין "קריאה בלבד"

מכיון שמקור הקבצים הוא התקליטור, הם מסומנים לקריאה בלבד. רצוי לשנות מאפיין זה בדרך זו:

1. דרך הסייר היכנס לתיקיה בדיסק, שבה נמצאים הקבצים שהעתקת.
2. סמן קובץ מסוים או את כולם על ידי Ctrl+A.
3. הצב את סמן העכבר מעל האזור המסומן ולחץ לחיצה ימנית בעכבר.

4. מהתפריט המקוצר בחר **מאפיינים**.

5. בטל את הסימון בתיבה **קריאה בלבד** (דאג שתיבה זו תהיה ריקה).

6. לחץ על **החל**.

7. לחץ על **אישור**.

מה עוד בתקליטור?

הוצאת **הוד-עמי** מפיצה תוכנות אלו כבונוס ללקוחות ההוצאה, ואינה מתיימרת לגבות תשלום עבור התוכניות המצורפות ו/או לתמוך בהם.



אזהרה:

השימוש בתקליטור זה הוא על אחריותו הבלעדית של המשתמש. המוצרים המותקנים בתקליטור זה מסופקים באחריות החברות המייצרות אותם. הוצאת **הוד-עמי** אינה אחראית, בכל צורה שהיא, לאופן ולטיב התוכנות המותקנות.

בכל שאלה לגבי תוכנה הנמצאת בתקליטור, יש לפנות למפתחי התוכנה (כל תוכנה בנפרד) כפי שמצוין בקבצי העזרה של התוכנה המדוברת.

הקבצים הם גרסאות **שיתופיות** (ShareWare) **וחופשיות** (FreeWare).

גרסת ShareWare מאפשרת לך, המשתמש, לבדוק את יעילות התוכנה ואת תאימותה לעבודה אותה אתה מבצע. אם נמצאה התוכנה מתאימה לצרכיך, עליך לשלם למפתחיה תשלום סמלי (לפי הרשום בקבצי העזרה של כל תוכנה ותוכנה בנפרד) כדי לקבל רישיון מלא לשימוש בה. קבלת רישיון לשימוש בתוכנה יפתח בפניך מיגוון אפשרויות שלא עמדו לרשותך בהפעלת גרסת ה-ShareWare.

Terra - מימד חדש בתצוגה

ים המלח ממעוף הציפור

Terra היא משפחת מוצרים של חברת סקייליין מערכות תוכנה, הבנויה על מנוע גרפי ייחודי ורב עוצמה, המאפשר תצוגה בזמן אמת של פני שטח בתלת-מימד ללא הגבלת פרטים וגודל על מחשבים ביתיים. מקור תמונת השטח הוא תצלומי לווין ותצלומי אוויר.

דרישות מערכת מינימליות:

- מעבד מסוג Pentium עם טכנולוגיית MMX
- זיכרון פנימי 32MB RAM
- זיכרון כרטיס מסך 2MB

- כונן תקליטורים במהירות x6
- מערכת הפעלה Windows 95/98
- התקליטור חייב להיות בכונן במהלך ההפעלה

מנוע Terra לוקח את התצוגה בזמן אמת לרמות חדשות שלא נודעו בעולם מחשבי PC. הוא מאפשר אין-סוף מצבי תצוגה: החל ממבט קרוב לפרטי פרטים ועד תצוגה רחוקה עד האופק. מנוע Terra עושה שימוש בטכנולוגיית MMX החדשנית ומאפשר איכויות שהיו נחלתם של מחשבים יקרים בלבד.

לפרטים נוספים על התוכנה ומפתחיה קרא בקובץ INFO בתיקיה Terra.

התקנת Terra

מומלץ להסיר גירסה קודמת של Terra, אם קיימת.

1. הכנס את התקליטור לכונן.
2. לחץ על לחצן **התחל** ובחר באפשרות **הפעלה**.
3. לחץ על לחצן **עיון**.
4. בחר בכונן התקליטורים בתיקיה **Terra** ובקובץ בשם **SETUP.EXE**.
5. לחץ על לחצן **פתח**.
6. לחץ על לחצן **אישור**.
7. פעל לפי ההוראות על המסך לפי הסדר (מימין לשמאל):

   Next, Yes, Next, Next, Next, Next, Finish.

ייתכן ובמהלך ההתקנה תתבקש להתקין רכיבי DirectX. אם הינך עובד במערכת Windows 95 עם ממשק עברית (לחצן **התחל**) יהיה עליך לשנות את ההגדרות האזוריות:

1. בחר בלחצן **התחל**, **הגדרות**, **לוח הבקרה**, **הגדרות אזוריות**.
2. במקום **עברית** בחר **אנגלית** (ארצות הברית).
3. בחר בלחצן **החל**.
4. לשאלה האם ברצונך לאתחל את המחשב מחדש? ענה **כן**.
5. עתה, יהיה עליך להתחיל את התקנת Terra מחדש.

הערה:

אם שינית את ההגדרות האזוריות - אל תשכח לחזור לעברית.

הפעלת Terra

לחץ על לחצן **התחל**, **תוכניות**, **Terra**, **TerraViewer**.

מקשים	**פעולה**
Shift+A	הגבר מהירות
Shift+Z	האט מהירות
מקשי חיצים	תנועה מעלה/מטה/ימינה/שמאלה

הוראות הפעלה מפורטות נמצאות בתפריט Help שבתוכנת Terra.



טיפ:

תוכל להאט את המהירות בעזרת Shift+Z לא רק למהירות אפס, אלא מתחת לזה. המשמעות היא ש... תטוס אחורה!!! שווה בדיקה!!!

התקליטור חייב להיות בכונן בעת הפעלת התוכנה.

התקנת תוכנת גלישה לאינטרנט Microsoft Internet Explorer 5

תוכנית ההתקנה מזהה את גרסת מערכת ההפעלה ומתקינה את גרסת הדפדפן הדרושה. מומלץ להסיר גירסה קודמת של Internet Explorer, אם קיימת.

1. הכנס את התקליטור לכונן.
2. לחץ על לחצן **התחל** ובחר באפשרות **הפעלה**.
3. לחץ על לחצן **עיון**.
4. בחר בכונן התקליטורים בתיקיה Software\IE5 ובקובץ בשם SETUP.EXE.
5. לחץ על לחצן **פתח**. לחץ על לחצן **אישור**.
6. פעל לפי ההוראות על המסך.



אזהרה:

לפני ביצוע שדרוג מ-Windows 95 ל-Windows 98 בעברית (זו בה התפריטים בעברית ולחצן התחל מימין שורת המשימות), יש להסיר את Internet Explorer 5. לאחר השדרוג ניתן לבצע התקנה מחדש של הגירסה המתאימה.

FontsPekan

קובץ זה יתקין במחשב 2 גופנים בעברית לשימושכם. בסיום ההתקנה יש לבצע את הפעולות הבאות:

1. לחץ על **התחל**, הצבע על **הגדרות**, ובחר ב**לוח הבקרה**.
2. לחץ לחיצה כפולה על הסמל **גופנים**.
3. פתח את תפריט **קובץ** ובחר באפשרות **התקנת גופן חדש**.
4. עבור לתיקיה **C:\FontsPekan**.
5. לחץ על לחצן **בחר הכל** (סה״כ יש בתיקיה 2 גופנים).
6. ודא שתיבת הסימון **העתק גופנים לתיקיית הגופנים** מסומנת.
7. לחץ **אישור**.
8. סגור את חלון התיקיה **Fonts**.
9. סגור את חלון **לוח הבקרה**.

כעת, מוכנים הגופנים לשימוש בכל התוכנות המותקנות במחשב שלך: Word, Excel, PowerPoint וגם בתוכנות גרפיות, כגון Paint Shop Pro ו-PhotoShop.

הגופנים נקראים Tml-JUMP ו-Tml-step ויופיעו בתחתית רשימת שמות הגופנים (בדרך כלל). הרי דוגמה שלהם:

Tml-step

אבגדהוזחטיכךלמםנןסעפףצץקרשת1234567890

Tml-JUMP

אבגדהוזחטיכךלמםנןסעפףצץקרשת1234567890

NETEX

במקום לרשום **http://www.hod-ami.co.il** פשוט רישמו **הוד-עמי** והנה אתם באתר ההוצאה.

במטרה להגיע לאתר מסוים באינטרנט, שכתובתו אינה ידועה, אנו משתמשים בדרך-כלל באחת משתי דרכים: ניחוש של כתובת האתר ו/או פנייה לאינדקס או למנוע חיפוש

שתי הפעולות הן מסורבלות וגוזלות זמן ואנרגיה מיותרים. ניחוש הכתובת מחייב הקלדה של הכתובת המלאה באנגלית בדיוק מושלם, והוא עשוי להיות הליך גוזל זמן, במיוחד כאשר לא מצליחים למצוא את האתר בניסיון ראשון או בכלל.

עם netex לא צריך לנחש כתובות או לגלוש למנועי חיפוש כדי להגיע לאתר מסוים ברשת! פשוט מקלידים את שם האתר בעברית בחלון הכתובת בדפדפן, ומגיעים אליו ישירות.

אפשרויות השימוש במערכת

גלישה ישירה לאתר על-פי שמו או על-פי נתונים הקשורים בו.

מקלידים בחלון הכתובת של הדפדפן שם של אתר או חברה או מילות מפתח הקשורות באתר (בכל סדר שהוא), ומגיעים אליו ישירות.

לדוגמה:

- מקלידים הוד-עמי או הוצאת הוד-עמי או ספרי מחשבים ומגיעים ישירות לאתר הוצאת הוד-עמי לספרי מחשבים.
- מקלידים **בנק דיסקונט** - ומגיעים ישירות לאתר של בנק דיסקונט.
- מקלידים **סלקום** או **052** - ומגיעים ישירות לאתר של חברת סלקום;
- מקלידים **עכבר העיר** - ומגיעים ישירות לאתר של העכבר;
- מקלידים **144** - ומגיעים למודיעין 144 של בזק;

התחברות למערכת הניתוב החדשה של ישראל

כל שעליכם לעשות כדי להתחבר ל-NETEX הוא להתקין תוכנה קלה וחכמה:

1. יש לסגור את כל התוכנות הפתוחות כולל הדפדפנים הפועלים.
2. מתוך סייר Windows הפעילו את הקובץ **netex100.exe** שבתיקיה X:\Software\NetEx.
3. פעלו בהתאם להוראות.

בזמן ההתקנה התוכנה מזהה את הדפדפן/ים שמותקנים במחשב, והיא תפעל עם כולם. מייד אחרי סיום ההתקנה תוכלו להתחיל לגלוש חכם ובעברית.

המערכת שקופה למשתמש, כלומר היא "מתלבשת" על הדפדפן הרגיל ואינה נראית כלל. אין צורך להפעיל אותה או לבצע פעולה כלשהי כדי להשתמש בה: פשוט מקלידים את שם האתר המבוקש בשדה הכתובת של הדפדפן, ומגיעים ישר אליו.

התוכנה אינה מפריעה לעבודה רגילה עם הדפדפן. היא נכנסת לפעולה רק כאשר מקלידים נתון שאינו כתובת אינטרנט רגילה (URL). כאשר תקלידו **www.hod-ami.co.il** תגלשו ישירות לאתר **הוצאת הוד-עמי**, בדיוק כפי שנהגתם לגלוש לפני התקנת התוכנה (ללא מעורבות המערכת). אך אם תרצו, תוכלו להקליד **הוד-עמי** בשדה כתובת ולהגיע במהירות לאותו אתר בדיוק.

תיקיה ראשית SoftWare (רשימה חלקית)

שם תוכנה	תיאור	קובץ הפעלה	מבצע
Adobe Acrobat	תוכנה לצפייה בקבצי pdf	Arme4ENU.exe	התקנה
Clean System	מחיקת קבצי dll שאין צורך בהם	ClnSys16.exe	התקנה
FontsPekan	גופנים בעברית	FontsPekan.exe	פריסה
ICQ	תוכנה לתקשורת אישית באינטרנט	ICQ99b.exe	התקנה
MIRC	תוכנת הצ'אט הפופולרית ביותר ברשת	mirc561t.exe	התקנה
NetEx	תוכנה המאפשרת גלישה בעבריץ	netex100.exe	התקנה
Paint Shop Pro 5.03	תוכנה ליצירת, עיצוב ועיבוד תמונות	Psp503ev.exe	התקנה
Power Toys	תוכניות שירות עבור Windows 95	PowerToy.exe	פריסה
WinAmp	תוכנה להשמעת קבצי MP3 (מוסיקה)	WinAmp25e.exe	התקנה
WinZip	תוכנית לפריסה/דחיסה של קבצים	WinZip7sr1.exe	התקנה
WordView	תוכנית לצפייה בקבצי doc	WordView.exe	התקנה

אינדקס עברי

S

V

W

א

ב

ג

ד

ה

ו

נ

ס

ע

ר

ש

ת

Index

Symbols

A

B

C

D

E

F

G

H

I

J - K

L

M

N

O

P

Q

R

S

T

U

V

W - Z

ת.ד. 6108 הרצליה 46160 **הזמנות באתר**
info@hod-ami.co.il www.**hod-ami**.co.il

הוצאת הוד-עמי ספרי מחשבים

קטלוג אוקטובר 2013 *הנחות ומבצעים באתר*

	כולל	ת. הוצאה	עמ'	מחיר*
אינטרנט - מפתחי אתרים/גרפיקה				
אמא, אבא - בניתי אתר באינטרנט		9/2003	256	**29**
הגדל את הכנסות העסק שלך באמצעות פרסום בגוגל Google AdWords		6/2010	300	249
מבוא לתכנות בסביבת אינטרנט – הכל כלול בספר אחד HTML + JS + ASP		3/2006	192	**9**
מבוא לתכנות בסביבת אינטרנט - מבוא ו- **HTML** - חלק 1 מתוך 3 - מהד' 3		6/2003	320	59
מבוא לתכנות בסביבת אינטרנט - **JavaScript** - חלק 2 מתוך 3		5/2002	240	49
מבוא לתכנות בסביבת אינטרנט - **ASP** - חלק 3 מתוך 3 - מהד' 3		3/2006	192	49
HTML5 המדריך לבניית אתרים, הדור הבא (חדש)		10/2012	352	149
HTML & CSS למפתחי אתרים באינטרנט - מהד' 5		4/2005	592	**79**
The **Java** Tutorial סדנת לימוד	CD	7/2001	768	179
JavaScript סדנת לימוד		2001	586	159
מדריך **ASP.NET MVC 4** (חדש)		7/2013	514	199
ASP.NET 3.5 סדנת לימוד בשפות C# ו-VB		10/2009	824	**99**
תכנות				
Code Complete – מדריך מעשי לפיתוח תוכנה (ברכישה ישירה) (חדש)		10/2013	288	139
לחפש באגים, מדריך מעשי לבודק תוכנה, מהד' 3 (ברכישה ישירה)		7/2011	350	169
Visual C# 3.0 סדנת לימוד		9/2008	656	**99**
ללמוד **C** - מהד' 3		2/2001	480	139
יסודות התכנות ב-**VBA** לתוכנת **Excel**, מהד' 3 (ברכישה ישירה)		2012	152	95
PC - חומרה, תוכנה ורשתות				
מדריך Hacking ואבטחת מידע, מהד' 2		9/2011	428	169
מדריך **חומרה ותוכנה** לטכנאי PC - מהד' 5 כולל עדכון 2011	CD	2/2011	752	189
Windows 7/8 נושאים מתקדמים (ברכישה ישירה. המחיר כולל משלוח)		(חדש)	140	49
מדריך **רשתות** לטכנאי PC ולמנהלי רשת - מהד' 4		4/2007	608	219
Windows				
Windows 8 מדריך למשתמש (חדש)		7/2013	438	129
Windows 7 צעד-אחר-צעד		1/2010	272	**39**
Windows XP והכרת המחשב האישי למתחילים		3/2003	208	**19**
גרפיקה				
Flash – ספר הדרכה ותרגילים (ברכישה ישירה)		2012	122	64
Illusatrator – ספר הדרכה ותרגילים (ברכישה ישירה)		2012	120	64
Photoshop צעד אחר צעד (צבע מלא, למתחילים), מהד' 3		3/2012	200	159
מדריך לתוכנת העיצוב והאנימציה **3ds max** (2 כרכים) (ברכישה ישירה)	CD	מהד' 5	1400	419

* מחיר מומלץ לצרכן כולל מע"מ (המחירים המודגשים במחיר מיוחד בהזמנה ישירה).

יש להיכנס לאתר כדי לבדוק מבצעים ומחירים מיוחדים

היכנס לאתר להתעדכן בספרים החדשים ומבצעים www.**hod-ami**.co.il

	כולל	הוצאה	עמ'	מחיר*
OFFICE 2010				
טבלאות ציר – ניתוח נתונים חכם (ברכישה ישירה)		2012	116	87
יסודות התכנות ב-**VBA** לתוכנת **Excel**, מהד' 3 (ברכישה ישירה)		2012	152	95
Word 2010 צעד אחר צעד		11/2010	160	69
Excel 2010 סדנת לימוד		11/2010	344	129
PowerPoint 2010 צעד אחר צעד		11/2010	202	69
Outlook 2010 צעד אחר צעד		11/2010	190	69
Access 2010 סדנת לימוד		11/2010	360	179
OFFICE 2007				
PowerPoint 2007 למתחילים		8/2007	240	99
Outlook 2007 למתחילים		1/2008	224	**19**
OFFICE 2003				
Word 2003 צעד אחר צעד		3/2004	208	**9**
Excel 2003 הסדרה הידידותית למתחילים		2/2004	160	49
PowerPoint 2003 הסדרה הידידותית למתחילים		2/2004	96	**9**
Outlook 2003 הסדרה הידידותית למתחילים		2/2004	96	**9**
Access 2003 הסדרה הידידותית למתחילים		2/2004	144	**9**
OFFICE XP				
Office XP צעד אחר צעד	CD	7/2001	576	**19**
Word XP הסדרה הידידותית למתחילים		7/2001	144	**9**
ניהול, כלכלה ושונות				
לחפש באגים, מדריך מעשי לבודק תוכנה, מהד' 3 (ברכישה ישירה)		7/2011	350	169
חקר ביצועים כשירות ללקוח		2/2010	236	92
ניהול ממוקד לעשות יותר עם מה שיש (כריכה קשה) - מהד' 4		9/2012	358	133
לי זה עולה יותר (תמחיר) (כריכה קשה) - מהד' 3		10/2006	368	129
מערכות מידע				
Oracle SQL יכולות מתקדמות (ברכישה ישירה)		7/2011	626	249
SAS (**Statistical Analysis System**) – ספר לימוד (ברכישה ישירה)		11/2010	256	169
בסיסי נתונים ושפת **SQL** – עקרונות ועיצוב (ברכישה ישירה)			648	149
ניתוח מערכות מידע כולל מתודולוגיית ה- **UML** (ברכישה ישירה)		11/2004	818	229
המדריך העברי השלם **UML** (ברכישה ישירה)		1/2009	346	329

* מחיר מומלץ לצרכן כולל מע"מ. קטלוג 10/2013 (המחירים המודגשים במחיר מיוחד ברכישה ישירה).

יש להיכנס לאתר כדי לבדוק מבצעים ומחירים מיוחדים

היכנס לאתר להתעדכן בספרים החדשים ומבצעים

www.**hod-ami**.co.il תוכן עניינים ופרקים לדוגמה